BGP for BIER-TE Path
draft-chen-idr-bier-te-path-03

Abstract

This document describes extensions to Border Gateway Protocol (BGP) for distributing a Bit Index Explicit Replication Traffic/Tree Engineering (BIER-TE) path. A new Tunnel Type for BIER-TE path is defined to encode the information about a BIER-TE path.

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] [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 22 June 2022.

Copyright Notice

Copyright (c) 2021 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ................................................. 3
1.1. Terminologies ........................................... 3
2. Overview of BGP for BIER-TE Path ............................ 4
  2.1. Example BIER-TE Topology with BGP ....................... 4
  2.2. Distributing Path to Ingress ............................ 5
3. Extensions to BGP ............................................. 6
  3.1. New SAFI and NLRI ....................................... 6
  3.2. New Tunnel Type for BIER-TE ............................. 7
  3.3. Path BitPositions Sub-TLV .............................. 8
  3.4. Path Name Sub-TLV ...................................... 9
  3.5. Traffic Description Sub-TLVs ........................... 10
4. Security Considerations ...................................... 11
5. Acknowledgements ............................................ 11
6. IANA Considerations ......................................... 11
  6.1. Existing Registry: SAFI Parameters ..................... 11
  6.2. Existing Registry: BGP TEA Tunnel Types ................ 12
  6.3. Existing Registry: BGP TEA sub-TLVs ................... 12
7. References .................................................. 12
  7.1. Normative References ................................... 12
  7.2. Informative References ................................ 13
Appendix A. Extensions to PMSI_TUNNEL Attribute ................. 13
  A.1. New Tunnel Type for BIER-TE ........................... 14
Authors’ Addresses ............................................. 14
1. Introduction

[I-D.ietf-bier-te-arch] introduces Bit Index Explicit Replication (BIER) Tree Engineering (BIER-TE). It is an architecture for per-packet stateless explicit point to multipoint (P2MP) multicast path/tree, which is called BIER-TE path, and based on the BIER architecture defined in [RFC8279].

A Bit-Forwarding Router (BFR) in a BIER-TE domain has a BIER-TE Bit Index Forwarding Table (BIFT). A BIER-TE BIFT on a BFR comprises a forwarding entry for a BitPosition (BP) assigned to each of the adjacencies of the BFR. If the BP represents a forward connected adjacency, the forwarding entry for the BP forwards the multicast packet with the BP to the directly connected BFR neighbor of the adjacency. If the BP represents a BFER (i.e., egress node) or say a local decap adjacency, the forwarding entry for the BP decapsulates the multicast packet with the BP and passes a copy of the payload of the packet to the packet’s NextProto within the BFR.

A Bit-Forwarding Ingress Router (BFIR) in a BIER-TE domain receives the information or instructions about which multicast flows/packets are mapped to which BIER-TE paths that are represented by BitPositions or say BitStrings. After receiving the information or instructions, the ingress node/router encapsulates the multicast packets with the BitPositions for the corresponding BIER-TE paths, replicates and forwards the packets with the BitPositions along the BIER-TE paths.

This document proposes some procedures and extensions to BGP for distributing a BIER-TE path to the Bit-Forwarding Ingress Router (BFIR) of the path. It specifies a way of encoding the information about a BIER-TE path in a BGP UPDATE message, which can be distributed to the BFIR of the path.

1.1. Terminologies

The following terminologies are used in this document.

BIER: Bit Index Explicit Replication.

BIER-TE: BIER Tree Engineering.

BFR: Bit-Forwarding Router.

BFIR: Bit-Forwarding Ingress Router.

BFER: Bit-Forwarding Egress Router.
BFR-id: BFR Identifier. It is a number in the range [1,65535].

BFR-NBR: BFR Neighbor.

BFR-prefix: An IP address (either IPv4 or IPv6) of a BFR.

BIRT: Bit Index Routing Table. It is a table that maps from the BFR-id (in a particular sub-domain) of a BFER to the BFR-prefix of that BFER, and to the BFR-NBR on the path to that BFER.

BIFT: Bit Index Forwarding Table.

P-tunnel: A multicast tunnel through the network of one or more SPs.

PMSI: Provider Multicast Service Interface. PMSI is an abstraction that represents a multicast service for carrying packets. A PMSI is instantiated via one or more P-tunnels.

I-PMSI A-D Route: Inclusive PMSI Auto-Discovery route.

S-PMSI A-D route: Selective PMSI Auto-Discovery route.

x-PMSI A-D route: A route that is either an I-PMSI A-D route or an S-PMSI A-D route.

2. Overview of BGP for BIER-TE Path

This section briefly the BGP for BIER-TE path and illustrates some details through a simple example BIER-TE topology.

2.1. Example BIER-TE Topology with BGP

An example BIER-TE domain topology using SDN controller with a BGP to distribute BIER-TE path is shown in Figure 1. There are 8 nodes/BFRs A, B, C, D, E, F, G and H in the domain. Nodes/BFRs A, H, E, F and D are BFIRs (i.e., ingress nodes) or BFERs (i.e., egress nodes). The controller has a BGP session with each of the edge nodes in the domain, including BFIRs (i.e., ingress nodes A, H, E, F and D), and each of the non edge nodes in the domain (i.e., nodes B, C and G). Note that some of connections and the BGP on each edge node are not shown in the figure.
2.2. Distributing Path to Ingress

This section describes how the SDN controller distributes a BIER-TE path to its ingress node.

There are two scenarios for distributing the BIER-TE path information. In the first scenario, the ingress node is directly connected to the controller. The path information should not be propagated beyond the ingress node. In the second scenario, the ingress node is not directly connected to the controller. The path information should be propagated throughout the domain until it reaches the ingress node.

Suppose that node A in Figure 1 wants to have a BIER-TE path from ingress node A to egress nodes H and F. The path satisfies a set of constraints. The controller obtains the request from an application or user configuration. It finds a BIER-TE path satisfying the constraints and distributes the path to ingress node A.

Figure 1: Example BIER-TE Topology with Controller

Nodes/BFRs D, F, E, H and A are BFERs (or BFIRs) and have local decap adjacency BitPositions 1, 2, 3, 4, and 5 respectively.

The BitPositions for the forward connected adjacencies are represented by i', where i is from 1 to 20.
If A is directly connected to the controller (e.g., as the example network in Figure 1), then the controller sends A the information about the path in a Update message in one of two ways. In one way, the controller sends each of its BGP peers, including the BGP peer running on node A, a Update message about the explicit path, with a route target matching the BGP identifier of ingress node A, and NO_ADVERTISE community. Ingress node A accepts this message from the controller and installs a forwarding entry for the BIER-TE path, but will not advertise it to any peer. All the other peers do not accept the message.

In another way, the controller sends A a Update message directly through the local session between them, but does not send the message to any other peers. The message contains the information about the path, a route target matching the BGP identifier of ingress node A and the NO_ADVERTISE community. Ingress node A accepts this message from the controller and installs a forwarding entry for the BIER-TE path, but will not advertise it.

If A is not directly connected to the controller, then the controller distributes the information about the explicit path to the ingress node A across the network. To achieve this, the controller advertises a BGP Update message to all its BGP peers, where the message contains the information about the path, a route target matching the BGP identifier of ingress node A, but does not have NO_ADVERTISE community. Each of the BGP peers advertises the received Update to its BGP neighbors according to normal BGP propagation rules. Eventually, ingress node A accepts this message and installs a forwarding entry for the BIER-TE path, which imports the packets to be transported by the path into the path.

3. Extensions to BGP

This section defines a new Tunnel Type (or say TLV) for BIER-TE path/tunnel under Tunnel Encapsulation Attribute and a new SAFI. This new SAFI and the existing AFI for IPv4/IPv6 pair uses a new NLRI for indicating a BIER-TE Path.

3.1. New SAFI and NLRI

A new SAFI, called BIER-TE path SAFI, is defined. Its codepoint (TBD1) is to be assigned by IANA. This new SAFI and the existing AFI for IPv4/IPv6 pair uses a new NLRI, which is defined as follows:
Where:

NLRI Length: 1 octet represents the length of NLRI. If the Length is anything other than 15 or 27, the NLRI is corrupt and the enclosing UPDATE message MUST be ignored.

Distinguisher: 4 octet value uniquely identifies the content/BIER-TE path.

Tunnel Identifier: 11/23 octet value contains:

* sub-domain-id (1 octet): It is id of the sub domain through which the BIER-TE tunnel crosses.

* BFR-id (2 octets): It is the BFR-id of the BFIR of the BIER-TE tunnel.

* Tunnel-ID (4 octets): It is a number uniquely identifying a BIER-TE tunnel within the BFIR and sub domain.

* BFR-prefix (4/16 octets): It is a BFR-prefix of the BFIR of the BIER-TE tunnel. It occupies 4 octets for IPv4 and 16 octets for IPv6.

3.2. New Tunnel Type for BIER-TE

A new Tunnel Type (or say TLV), called BIER-TE Path or Tunnel, is defined under Tunnel Encapsulation Attribute in [RFC9012]. Its codepoint is to be assigned by IANA. This new TLV with a number of new sub-TLVs encodes the information about a BIER-TE path.

The structure encoding the information about a BIER-TE path is shown below.
Attributes:
  Tunnel Encaps Attribute (23)
  Tunnel Type (TBD2): BIER-TE Path
  Path BitPositions sub-TLV
  Path Name sub-TLV
  Traffic Description sub-TLV

Where:

* Tunnel Type (TBD2) is to be assigned by IANA.
* Path BitPositions sub-TLV encodes the bit positions of the BIER-TE path.
* Path Name sub-TLV encodes the name of a BIER-TE path.
* Traffic Description sub-TLV encodes the multicast traffic that is transported by the BIER-TE path.

3.3. Path BitPositions Sub-TLV

The bit positions of a BIER-TE path are encoded in a Path BitPositions sub-TLV. The format of the sub-TLV is illustrated below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (TBD3) |        Length (variable)      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    SI-Len     | BitStringLen  | sub-domain-id |     MT-ID     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                BIFT-id-1              |  RSV  |     SI-1      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                BitString-1                        ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                BIFT-id-n              |  RSV  |     SI-n      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                BitString-n                        ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Path BitPositions Sub-TLV Format
Type: Its value (TBD3) is to be assigned by IANA.

Length: It is variable.

Reserved/RSV: MUST be set to zero by the sender and MUST be ignored by the receiver.

SI-Len (SI Length) - 8 bits: The length in bits of the SI field.

BitStringLen (Bit String Length) - 8 bits: The length in bits of the BitString field according to [RFC8296]. If \( k \) is the length of the BitString, the value of BitStringLen is \( \log_2(k) - 5 \). For example, BitStringLen = 1 indicates \( k = 64 \), BitStringLen = 7 indicates \( k = 4096 \).

sub-domain-id: Unique value identifying the BIER sub-domain within the BIER domain.

MT-ID: Multi-Topology ID identifying the topology that is associated with the BIER sub-domain.

\(<\text{BIFT-id, SI, BitString}>\) tuple: Each tuple \(<\text{BIFT-id-i, SI-i, BitString-i}>\) (\( i = 1, 2, ..., n \)) represents/encodes a set of bit positions on the BIER-TE path with a BIFT ID. All the tuples in the sub-TLV represent/encode the BIER-TE path (i.e., all the bit positions of the BIER-TE path).

3.4. Path Name Sub-TLV

The name of a BIER-TE path is encoded in a Path Name sub-TLV. The format of the sub-TLV is illustrated below.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type (TBD4) |        Length (variable)      |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                        Path Name String                     //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: Path Name Sub-TLV Format

Type: Its value (TBD4) is to be assigned by IANA.

Length: It is variable.

Reserved: MUST be set to zero by the sender and MUST be ignored by
the receiver.

Path Name String: It represents/encodes the name of the BIER-TE path in a string of chars.

3.5. Traffic Description Sub-TLVs

A Traffic Description Sub-TLV describes the traffic to be imported into a BIER-TE path. Two Traffic Description Sub-TLVs are defined. They are multicast traffic sub-TLVs for IPv4 and IPv6.

The multicast traffic sub-TLVs for IPv4 and IPv6 are shown in Figure 5 and Figure 6 respectively.

```
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 (TBD5) |             Length            |   Reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Reserved           |S|G|  Src Mask Len | Grp Mask Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Source Address (up to 4 bytes)                 ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Group Multicast Address (up to 4 bytes)            ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 5: Multicast Traffic for IPv4 Sub-TLV

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 (TBD6) |             Length            |   RESERVED    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Reserved           |S|G|  Src Mask Len | Grp Mask Len  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Source Address                         ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                       (up to 16 bytes)                        ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Group multicast Address                     ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜                       (up to 16 bytes)                        ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Figure 6: Multicast Traffic for IPv6 Sub-TLV
```
The address fields and address mask lengths of the two Multicast Traffic sub-TLVs contain source and group prefixes for matching against packets noting that the two address fields are up to 32 bits for an IPv4 Multicast Traffic and up to 128 bits for an IPv6 Multicast Traffic.

The Reserved field MUST be set to zero and ignored on receipt.

Two bit flags (S and G) are defined to describe the multicast wildcarding in use. If the S bit is set, then source wildcarding is in use and the values in the Source Mask Length and Source Address fields MUST be ignored. If the G bit is set, then group wildcarding is in use and the values in the Group Mask Length and Group multicast Address fields MUST be ignored. The G bit MUST NOT be set unless the S bit is also set: if a Multicast Traffic sub-TLV is received with S bit = 0 and G bit = 1 the receiver MUST respond with an error (Malformed Multicast Traffic).

The three multicast mappings may be achieved as follows:

(S, G): S bit = 0, G bit = 0, the Source Address and Group multicast Address prefixes are both used to define the multicast traffic.

(*, G): S bit = 1, G bit = 0, the Group multicast Address prefix is used to define the multicast traffic, but the Source Address prefix is ignored.

(*, *): S bit = 1, G bit = 1, the Source Address and Group multicast Address prefixes are both ignored.

4. Security Considerations

Protocol extensions defined in this document do not affect the BGP security other than those as discussed in the Security Considerations section of [RFC9012].

5. Acknowledgements

The authors of this document would like to thank Tony Przygienda, Susan Hares, and Jeffrey Zhang for their comments.

6. IANA Considerations

6.1. Existing Registry: SAFI Parameters

This document requests assigning a new SAFI in the registry "Subsequent Address Family Identifiers (SAFI) Parameters" as follows:
### Existing Registry: BGP TEA Tunnel Types

This document requests assigning a new Tunnel-Type in the registry "BGP Tunnel Encapsulation Attribute Tunnel Types" as follows:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD2(16 suggested)</td>
<td>BIER-TE Tunnel/Path</td>
<td>This document</td>
</tr>
</tbody>
</table>

### Existing Registry: BGP TEA sub-TLVs

This document requests assigning a few of new sub-TLVs in the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" as follows:

<table>
<thead>
<tr>
<th>Code Point</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD3(16 suggested)</td>
<td>Path BitPositions</td>
<td>This document</td>
</tr>
<tr>
<td>TBD4(17 suggested)</td>
<td>Path Name</td>
<td>This document</td>
</tr>
<tr>
<td>TBD5(18 suggested)</td>
<td>IPv4 Multicast Traffic</td>
<td>This document</td>
</tr>
<tr>
<td>TBD6(19 suggested)</td>
<td>IPv6 Multicast Traffic</td>
<td>This document</td>
</tr>
</tbody>
</table>

### References

#### Normative References


7.2. Informative References


Appendix A. Extensions to PMSI_TUNNEL Attribute

This section defines a new Tunnel Type (or TLV) for BIER-TE path/tunnel under the PMSI_TUNNEL Attribute (PTA) defined in [RFC6514]. It describes a couple of new sub-TLVs encoding the information about a BIER-TE path.
A.1.  New Tunnel Type for BIER-TE

The PMSI Tunnel attribute carried by an x-PMSI A-D route identifies P-tunnel for PMSI. For the PTA with Tunnel Type BIER-TE, the PTA is constructed by the SDN controller and distributed to the ingress node of the BIER-TE tunnel.

The format of the PMSI_TUNNEL Attribute with the new Tunnel Type (TBD) for BIER-TE is shown in Figure 7.

For BIER-TE tunnel/path, the fields in the PTA are set as follows:

- **Tunnel Type**: It is set to be TBD, indicating BIER-TE tunnel.


- **sub-TLVs**: It contains a Path BitPositions sub-TLV encoding an explicit BIER-TE path. It may include a Path Name sub-TLV for the name of the BIER-TE path.

- **Others**: The other fields are set according to [RFC6514].
Huaimo Chen  
Futurewei  
Boston, MA,  
United States of America  
Email: huaimo.chen@futurewei.com

Mike McBride  
Futurewei  
Email: michael.mcbride@futurewei.com

Ran Chen  
ZTE Corporation  
Email: chen.ran@zte.com.cn

Gyan S. Mishra  
Verizon Inc.  
13101 Columbia Pike  
Silver Spring, MD 20904  
United States of America  
Phone: 301 502-1347  
Email: gyan.s.mishra@verizon.com

Aijun Wang  
China Telecom  
Beiqijia Town, Changping District  
Beijing  
102209  
China  
Email: wangaj3@chinatelecom.cn

Yisong Liu  
China Mobile  
Email: liuyisong@chinamobile.com
Yanhe Fan
Casa Systems
United States of America
Email: yfan@casa-systems.com

Boris Khasanov
Yandex LLC
Moscow
Email: bhassanov@yahoo.com

Lei Liu
Fujitsu
United States of America
Email: liulei.kddi@gmail.com

Xufeng Liu
Volta Networks
McLean, VA
United States of America
Email: xufeng.liu.ietf@gmail.com
BGP-LS Extensions for Scalable Segment Routing based Enhanced VPN
draft-dong-idr-bgpls-sr-enhanced-vpn-04

Abstract

Enhanced VPN (VPN+) aims to provide enhanced VPN services to support some applications’ needs of enhanced isolation and stringent performance requirements. VPN+ requires integration between the overlay VPN connectivity and the resources and characteristics provided by the underlay network. A Virtual Transport Network (VTN) is a virtual underlay network which can be used to support one or a group of VPN+ services. In the context of network slicing, a VTN could be instantiated as a network resource partition (NRP).

This document specifies the BGP-LS mechanisms with necessary extensions to advertise the information of scalable Segment Routing (SR) based NRPs to a centralized network controller. Each NRP can have a customized topology and a set of network resources allocated from the physical network. Multiple NRPs may share the same topology, and multiple NRPs may share the same set of network resources on specific network segments. This allows flexible combination of network topology and network resource attributes to build a large number of NRPs with a relatively small number of logical topologies. The proposed mechanism is applicable to both segment routing with MPLS data plane (SR-MPLS) and segment routing with IPv6 data plane (SRv6).

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 5 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction .................................................. 3
2. Advertisement of NRP Definition .............................. 4
3. Advertisement of NRP Topology Attribute ..................... 5
   3.1. Intra-domain Topology Advertisement ..................... 6
       3.1.1. MTR based Topology Advertisement .................. 6
       3.1.2. Flex-Algo based Topology Advertisement .......... 7
   3.2. Inter-Domain Topology Advertisement .................... 8
       3.2.1. NRP IDs TLV ......................................... 9
3. Advertisement of NRP Resource Attribute ..................... 10
   4.1. Option 1: L2 Bundle based Approach ..................... 11
   4.2. Option 2: Per-NRP Link TE Attributes ................... 12
4. Advertisement of NRP specific Data Plane Identifiers ....... 13
   5.1. NRP-specific SR-MPLS SIDs ............................. 13
       5.1.1. NRP-specific Prefix-SID TLV ..................... 13
       5.1.2. NRP-specific Adj-SID TLV ......................... 14
   5.2. NRP-specific SRv6 SIDs .................................. 15
       5.2.1. NRP-specific SRv6 Locators and End SIDs .......... 15
       5.2.2. NRP-specific SRv6 End.X SID .................... 16
   5.3. Dedicated NRP ID in Data Plane .......................... 17
5. Security Considerations ....................................... 17
1. Introduction

Enhanced VPN (VPN+) is an enhancement to VPN services to support the needs of new applications, particularly the applications that are associated with 5G services. These applications require enhanced isolation and have more stringent performance requirements than that can be provided with traditional overlay VPNs. These properties require integration between the underlay and the overlay networks. [I-D.ietf-teas-enhanced-vpn] specifies the framework of enhanced VPN and describes the candidate component technologies in different network planes and layers. An enhanced VPN can be used for 5G network slicing, and will also be of use in more generic scenarios.

To meet the requirement of enhanced VPN services, a number of virtual underlay networks need to be created, each with a subset of the underlay network topology and a set of network resources allocated to meet the requirement of a specific VPN+ service or a group of VPN+ services. Such a virtual underlay network is called Virtual Transport Network (VTN) in [I-D.ietf-teas-enhanced-vpn]. [I-D.ietf-teas-ietf-network-slices] introduces the concept Network Resource Partition (NRP) as a set of network resources that are available to carry traffic and meet the SLOs and SLEs. In order to allocate network resources to an NRP, the NRP is associated with a network topology to define the set of links and nodes. Thus VTN and NRP are similar concepts, and NRP can be seen as an instantiation of VTN in the context of network slicing. For clarity, the rest of this document uses NRP in the description of the proposed mechanisms and protocol extensions.
[I-D.ietf-spring-resource-aware-segments] introduces resource-awareness to Segment Routing (SR) [RFC8402] by associating existing type of SIDs with network resource attributes (e.g. bandwidth, processing or storage resources). These resource-aware SIDs retain their original functionality, with the additional semantics of identifying the set of network resources available for the packet processing action. [I-D.ietf-spring-sr-for-enhanced-vpn] describes the use of resource-aware segments to build SR based NRPs. To allow the network controller and network nodes to perform NRP-specific explicit path computation and/or shortest path computation, the group of resource-aware SIDs allocated by network nodes to each NRP and the associated topology and resource attributes need to be distributed in the control plane.

When an NRP spans multiple IGP areas or multiple Autonomous Systems (ASes), BGP-LS is needed to advertise the NRP information in each IGP area or AS to the network controller, so that the controller could use the collected information to build the view of inter-area or inter-AS SR NRPs.

This document describes BGP-LS [RFC7752] based mechanism with necessary extensions to advertise the topology and resource attribute of inter-area and inter-domain SR based NRPs. Each NRP can have a customized topology and a set of network resources allocated. Multiple NRPs may share the same topology, and some of the NRPs may share the same set of network resources on specific network segments. This allows flexible combination of network topology and network resource attributes to build a large number of NRPs with a relatively small number of logical topologies. The definition of NRP is advertised as a node attribute using BGP-LS. The associated network topology and resources attributes of a NRP are advertised as link attributes using BGP-LS.

2. Advertisement of NRP Definition

According to [I-D.ietf-teas-ietf-network-slices], an NRP consists of a set of dedicated or shared network resources, and is associated with a customized network topology. Thus a NRP can be defined as the combination of a set of network attributes, which include the topology attribute and other attributes, such as the associated network resources.

The Network Resource Partition Definition (NRPD) TLV is a new TLV of the optional BGP-LS Attribute which is associated with the node NLRI.

The format of NRPD TLV is as follows:
Where:

* Type: To be assigned by IANA.

* Length: the length of the value field of the TLV. It is variable dependent on the included Sub-TLVs.

* NRP ID: A global significant 32-bit identifier which is used to identify an NRP.

* MT-ID: 16-bit identifier which contains the multi-topology identifier of the IGP topology.

* Algorithm: 8-bit identifier which indicates the algorithm which applies to this virtual transport network. It can be either a normal algorithm in [RFC8402] or a Flex-Algorithm [I-D.ietf-lsr-flex-algo].

* Flags: 8-bit flags. Currently all the flags are reserved for future use. They SHOULD be set to zero on transmission and MUST be ignored on receipt.

* Sub-TLVs: optional sub-TLVs to specify the additional attributes of an NRP. Currently no sub-TLV is defined in this document.

3. Advertisement of NRP Topology Attribute

[I-D.dong-lsr-sr-enhanced-vpn] describes the IGP mechanisms to distribute the topology attributes of SR based NRPs. This section describes the BGP-LS mechanism to distribute both the intra-domain and inter-domain topology attributes of SR based NRPs.
3.1. Intra-domain Topology Advertisement

The intra-domain topology attribute of an NRP can be determined by the MT-ID and/or the algorithm ID included in the NRP definition. In practice, it could be described using two optional approaches.

The first approach is to use Multi-Topology Routing (MTR) [RFC4915] [RFC5120] with the segment routing extensions to advertise the topology associated with the SR based NRPs. Different algorithms MAY be used to further specify the computation algorithm or the metric type used for path computation within the topology. Multiple NRPs can be associated with the same <topology, algorithm> tuple, and the IGP computation with the <topology, algorithm> tuple can be shared by these NRPs.

The second approach is to use Flex-Algo [I-D.ietf-lsr-flex-algo] to describe the topological constraints of SR based NRPs on a network topology (e.g. the default topology). Multiple NRPs can be associated with the same Flex-Algo, and the IGP computation result with this Flex-Algo can be shared.

This section describes the two optional approaches to advertise the intra-domain topology of an NRP using BGP-LS.

3.1.1. MTR based Topology Advertisement

In section 4.2.2.1 of [I-D.ietf-idr-rfc7752bis], Multi-Topology Identifier (MT-ID) TLV is defined, which can contain one or more IS-IS or OSPF Multi-Topology IDs. The MT-ID TLV MAY be present in a Link Descriptor, a Prefix Descriptor, or the BGP-LS Attribute of a Node NLRI.

[RFC9085] defines the BGP-LS extensions to carry the segment routing information using TLVs of BGP-LS Attribute. When MTR is used with SR-MPLS data plane, topology-specific prefix-SIDs and topology-specific Adj-SIDs can be carried in the BGP-LS Attribute associated with the prefix NLRI and link NLRI respectively, the MT-ID TLV is carried in the prefix descriptor or link descriptor to identify the corresponding topology of the SIDs.

[I-D.ietf-idr-bgppls-srv6-ext] defines the BGP-LS extensions to advertise SRv6 segments along with their functions and attributes. When MTR is used with SRv6 data plane, the SRv6 Locator TLV is carried in the BGP-LS Attribute associated with the prefix-NLRI, the MT-ID TLV can be carried in the prefix descriptor to identify the corresponding topology of the SRv6 Locator. The SRv6 End.X SIDs are carried in the BGP-LS Attribute associated with the link NLRI, the MT-ID TLV can be carried in the link descriptor to identify the
corresponding topology of the End.X SIDs. The SRv6 SID NLRI is
defined to advertise other types of SRv6 SIDs, in which the SRv6 SID
Descriptors can include the MT-ID TLV so as to advertise topology-
specific SRv6 SIDs.

[I-D.ietf-idr-rfc7752bis] also defines the rules of the usage of MT-
ID TLV:

"In a Link or Prefix Descriptor, only a single MT-ID TLV containing
the MT-ID of the topology where the link or the prefix is reachable
is allowed. In case one wants to advertise multiple topologies for a
given Link Descriptor or Prefix Descriptor, multiple NLRIs MUST be
generated where each NLRI contains a single unique MT-ID."

Editor’s note: the above rules indicates that only one MT-ID is
allowed to be carried the Link or Prefix descriptors. When a link or
prefix needs to be advertised in multiple topologies, multiple NLRIs
needs to be generated to report all the topologies the link or prefix
participates in, together with the topology-specific segment routing
information and link attributes. This may increase the number of BGP
Updates needed for advertising MT-specific topology attributes, and
may introduce additional processing burden to both the sending BGP
speaker and the receiving network controller. When the number of
topologies in a network is not a small number, some optimization may
be needed for the reporting of multi-topology information and the
associated segment routing information in BGP-LS. Based on the WG’s
opinion, this will be elaborated in a future version.

3.1.2. Flex-Algo based Topology Advertisement

The Flex-Algo definition [I-D.ietf-lsr-flex-algo] can be used to
describe the calculation-type, the metric-type and the topological
constraints for path computation on a network topology. As specified
in [I-D.dong-lsr-sr-enhanced-vpn], the topology of a NRP can be
determined by applying Flex-Algo constraints on a network topology.

BGP-LS extensions for Flex-Algo [I-D.ietf-idr-bgp-ls-flex-algo]
provide the mechanisms to advertise the Flex-Algo definition
information. BGP-LS extensions for SR-MPLS [RFC9085] and SRv6
[I-D.ietf-idr-bgppls-srv6-ext] provide the mechanism to advertise the
algorithm-specific segment routing information.
In [RFC9085], algorithm-specific prefix-SIDs can be advertised in BGP-LS attribute associated with Prefix NLRI. In [I-D.ietf-idr-bgppl-srv6-ext], algorithm-specific SRv6 Locators can be advertised in BGP-LS Attribute associated with the corresponding Prefix NLRI, and algorithm-specific End.X SID can be advertised in BGP-LS Attribute associated with the corresponding Link NLRI. Other types of SRv6 SIDs can also be algorithm-specific and are advertised using the SRv6 SID NLRI.

3.2. Inter-Domain Topology Advertisement

In some network scenarios, an NRP which spans multiple areas or ASes needs to be created. The multi-domain NRP could have different inter-domain connectivity, and may be associated with different set of network resources in each domain and also on the inter-domain links. In order to build the multi-domain NRPs using segment routing, it is necessary to advertise the topology and resource attribute of NRP on the inter-domain links and the associated BGP Peering SIDs.

[RFC9086] and [I-D.ietf-idr-bgppl-srv6-ext] defines the BGP-LS extensions for advertisement of BGP topology information between ASes and the associated BGP Peering Segment Identifiers. Such information could be used by a network controller for the computation and instantiation of inter-AS traffic engineering SR paths.

Depending on the network scenarios and the requirement of inter-domain NRPs, different mechanisms can be used to specify the inter-domain connections of NRPs.

* One EBGP session between two ASes can be established over multiple underlying links. In this case, different underlying links can be used for different inter-domain NRPs which requires link isolation between each other. In another similar case, the EBGP session is established over a single link, while the network resource (e.g. bandwidth) on this link can be partitioned into different pieces, each of which can be considered as a virtual member link. In both cases, different BGP Peer-Adj-SIDs SHOULD be allocated to each underlying physical or virtual member link, and ASBRs SHOULD advertise the NRP identifier associated with each BGP Peer-Adj-SID.
For inter-domain connection between two ASes, multiple EBGP sessions can be established between different set of peering ASBRs. It is possible that some of these BGP sessions are used for one inter-domain NRP, while some other BGP sessions are used for another inter-domain NRP. In this case, different BGP peer-node-SIDs are allocated to each BGP session, and ASBRs SHOULD advertise the NRP identifier associated with each BGP Peer-node-SIDs.

At the AS-level topology, different inter-domain NRPs may have different inter-domain connectivity. Different BGP Peer-Set-SIDs can be allocated to represent the groups of BGP peers which can be used for load-balancing in each inter-domain NRP.

In network scenarios where the MT-ID or Flex-Algo is used consistently in multiple areas or ASes covered by a NRP, the approaches to advertise topology-specific BGP peering SIDs are described as below:

* Using MT-based mechanism, the topology-specific BGP peering SIDs can be advertised with the MT-ID associated with the NRP carried in the corresponding link NLRI. This can be supported with the existing mechanisms defined in [RFC7752][RFC9086] and [I-D.ietf-idr-bgpls-srv6-ext].

* Using Flex-Algo based mechanism, the topology-specific BGP peering SIDs can be advertised together with the Admin Group (color) of the corresponding Flex-Algo in the BGP-LS attribute.

In network scenarios where consistent usage of MT-ID or Flex-Algo among multiple ASes can not be expected, then the global-significant NRP-ID can be used to define the AS level topologies. Within each domain, the MT or Flex-Algo based mechanism could still be used for topology advertisement.

3.2.1. NRP IDs TLV

A new NRP IDs TLV is defined to describe the identifiers of one or more NRPs an intra-domain or inter-domain link belongs to. It can be carried in BGP-LS attribute which is associated with a Link NLRI, or it could be carried as a sub-TLV in the L2 Bundle Member Attribute TLV.

The format of NRP IDs TLV is as below:
4. Advertisement of NRP Resource Attribute

[I-D.dong-lsr-sr-enhanced-vpn] specifies the optional mechanism to advertise the resource information associated with each NRP. One approach is to use the L2 bundle mechanism [RFC8668] to advertise the set of link resources allocated to an NRP as a L2 physical or virtual member link. Another approach is to advertise the set of network resources as per NRP link TE attributes. This section defines the corresponding BGP-LS extensions for both approaches.

Two new TLVs are defined to carry the NRP ID and the link attribute flags of either a Layer-3 link or the L2 bundle member links. The NRP ID TLV is defined in section 3.2.1 of this document, and a new Link Attribute Flags TLV is defined in this section. The TE attributes of each Layer 3 link or the L2 bundle member link, such as the bandwidth and the SR SIDs, can be advertised using the mechanism as defined in [RFC9085][RFC9086] and [I-D.ietf-idr-bgpls-srv6-ext].
4.1. Option 1: L2 Bundle based Approach

On an Layer-3 interface, each NRP can be allocated with a subset of link resources (e.g. bandwidth). A subset of link resources may be dedicated to an NRP, or may be shared by a group of NRPs. Each subset of link resource can be instantiated as a virtual layer-2 member link under the Layer-3 interface, and the Layer-3 interface is considered as a virtual Layer-2 bundle. The Layer-3 interface may also be a physical Layer 2 link bundle, in this case a subset of link resources allocated to an NRP may be provided by one of the physical Layer-2 member links.

The NRP ID TLV defined in section 3.2.1 of this document is used to carry the NRP IDs associated with the L2 bundle member links. The TE attributes of the L2 bundle member links, such as the maximum link bandwidth, and the SR SIDs, can be advertised using the mechanism as defined in [RFC9085][RFC9086] and [I-D.ietf-idr-bgpls-srv6-ext].

A new Link attribute Flags TLV is defined to specify the characteristics of a link. It can be carried in BGP-LS attribute which is associated with a Link NLRI, or it could be carried as a sub-TLV in the L2 Bundle Member Attribute TLV. The format of the sub-TLV is as below:

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

Where:

- **Type**: TBD
- **Length**: 4 octets.
- **Flags**: 16-bit flags. This field is consistent with the Flag field in IS-IS Link Attribute sub-TLV in [RFC5029]. In addition to the flags defined in [RFC5029], a new Flag "E" is defined in this document.
  - Link excluded from load balancing. When the flag is set, it indicates this link is only used for the associated NRPs.
4.2. Option 2: Per-NRP Link TE Attributes

An Layer-3 interface can participate in multiple NRPs, each of which is allocated with a subset of the resources of the interface. For each NRP, the associated resources can be described using per-NRP TE attributes. A new NRP-specific TE attribute TLV is defined to advertise the link attributes associated with an NRP. This sub-TLV MAY be carried in the BGP-LS Attribute associated with a Link NLRI.

The format of the NRP-specific TE attribute TLV is shown as below:

```
<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flags</td>
<td>Reserved</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>NRP IDs Sub-TLV</td>
<td></td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>Other Sub-TLVs</td>
<td></td>
</tr>
</tbody>
</table>
```

Where:

* **Type**: To be assigned by IANA.

* **Length**: The length of the value field of the TLV. It is variable dependent on the length of the Sub-TLVs field.

* **Flags**: 16-bit flags. All the 16 bits are reserved for future use, which SHOULD be set to 0 on transmission and MUST be ignored on receipt.

* **Reserved**: 16-bit field reserved for future use, SHOULD be set to 0 on transmission and MUST be ignored on receipt.

The NRP IDs TLV as defined in section 3.2.1 is used as the NRP IDs Sub-TLV in the per-NRP Link TE Attribute TLV.

Other Sub-TLVs are optional and can be used to carry the TE attributes associated with the NRPs. The existing Link TE Attribute TLVs as defined in [I-D.ietf-idr-rfc7752bis] can be reused as sub-TLVs here. New sub-TLVs may be defined in the future.
5. Advertisement of NRP specific Data Plane Identifiers

In network scenarios where each NRP is associated with an independent topology or Flex-Algo, the topology or Flex-Algo specific SR SIDs or Locators could be used to identify the NRP in data plane, so that the set of network resources associated with the NRP can be determined. In network scenarios where multiple NRPs share the same topology or Flex-Algo, additional data plane identifiers are needed to identify different NRPs.

This section describes the mechanisms to advertise the NRP identifiers with different data plane encapsulations.

5.1. NRP-specific SR-MPLS SIDs

With SR-MPLS data plane, the NRP identifier can be implicitly determined by the SR SIDs associated with the NRP. Each node SHOULD allocate NRP-specific Prefix-SIDs for each NRP it participates in. Similarly, NRP-specific Adj-SIDs MAY be allocated for each link which participates in the NRP.

5.1.1. NRP-specific Prefix-SID TLV

A new NRP-specific Prefix-SID TLV is defined to advertise the relationship between the prefix-SID and its associated NRP. It is derived from NRP-specific Prefix-SID sub-TLV of IS-IS [I-D.dong-lsr-sr-enhanced-vpn]. The format of the sub-TLV is as below:

```
0                   1                   2                   3
+---------------+---------------+---------------+---------------+
|             Type|             Length|
+---------------+---------------+---------------+---------------+
|            Flags|            Reserved|
+---------------+---------------+---------------+---------------+
|                    NRP ID|
|+----------------------------+
|                     SID/Index/Label(Variable)|
```

Where:

* Type: TBD

* Length: The length of the value field of the sub-TLV. It is variable dependent on the length of the SID/Index/Label field.
* Flags: 16-bit flags. The high-order 8 bits are the same as in the Prefix-SID sub-TLV defined in [RFC8667]. The lower-order 8 bits are reserved for future use, which SHOULD be set to 0 on transmission and MUST be ignored on receipt.

* Reserved: 16-bit field reserved for future use, SHOULD be set to 0 on transmission and MUST be ignored on receipt.

* NRP ID: A 32-bit local identifier to identify the NRP this prefix-SID is associated with.

* SID/Index/Label: The same as defined in [RFC8667].

One or more of NRP-specific Prefix-SID TLVs MAY be carried in BGP-LS attribute of the associated Prefix NLRI. The MT-ID in the Prefix descriptors SHOULD be the same as the MT-ID in the definition of the NRP.

5.1.2. NRP-specific Adj-SID TLV

A new NRP-specific Adj-SID TLV is defined to advertise between the Adj-SID and its associated NRP. It is derived from NRP specific Adj-SID sub-TLV of IS-IS [I-D.dong-lsr-sr-enhanced-vpn]. The format of the sub-TLV is as below:

```
<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+-------------------------------------------------+</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Where:

* Type: TBD

* Length: The length of the value field of the sub-TLV. It is variable dependent on the length of the SID/Index/Label field.

* Flags: 16-bit flags. The high-order 8 bits are the same as in the Adj-SID sub-TLV defined in [RFC8667]. The lower-order 8 bits are reserved for future use, which SHOULD be set to 0 on transmission and MUST be ignored on receipt.
5.2. NRP-specific SRv6 SIDs

5.2.1. NRP-specific SRv6 Locators and End SIDs

With SRv6 data plane, the NRP identifier can be implicitly or explicitly determined using the SRv6 Locators associated with the NRP, this is to ensure that all network nodes (including both the SRv6 End nodes and Transit nodes) can identify the NRP to which a packet belongs. Network nodes SHOULD allocate NRP-specific Locators for each NRP it participates in. The NRP-specific Locators are used as the covering prefix of NRP-specific SRv6 End SIDs, End.X SIDs and other types of SIDs.

Each NRP-specific SRv6 Locator MAY be advertised in a separate Prefix NLRI. If multiple NRPs share the same topology/algorithm, the topology/algorithm specific Locator is the covering prefix of a group of NRP-specific Locators. Then the advertisement of NRP-specific locators can be optimized to reduce the amount of information advertised in the control plane.

A new NRP locator-block sub-TLV under the SRv6 Locator TLV is defined to advertise a set of sub-blocks which follows the topology/algorithm specific Locator. Each NRP locator-block value is assigned to one of the NRPs which share the same topology/algorithm.
5.2.2. NRP-specific SRv6 End.X SID

The SRv6 End.X SIDs are advertised in the BGP-LS attribute with Link NLRI. In order to distinguish the End.X SIDs which belong to different NRPs, a new "NRP ID Sub-TLV" is introduced under the SRv6 End.X SID TLV and SRv6 LAN End.X SID TLV defined in [I-D.ietf-idr-bgpls-srv6-ext]. Its format is shown as below:
Where:

* Type: TBD.
* Length: the length of the Value field of the TLV. It is set to 4.
* NRP ID: A 32-bit global identifier to identify the NRP this End.X SID is associated with.

5.3. Dedicated NRP ID in Data Plane

As the number of NRPs increases, with the mechanism described in [I-D.ietf-spring-sr-for-enhanced-vpn], the number of SR SIDs and SRv6 Locators allocated for different NRPs would also increase. In network scenarios where the number of SIDs or Locators becomes a concern, some data plane optimization may be needed to reduce the amount of SR SIDs and Locators allocated. As described in [I-D.dong-teas-nrp-scalability], one approach is to decouple the data plane identifiers used for topology based forwarding and the identifiers used for the NRP-specific processing. Thus a new data plane global NRP-ID could be introduced and encapsulated in the packet. One possible encapsulation of NRP-ID in IPv6 data plane is proposed in [I-D.dong-6man-enhanced-vpn-vtn-id]. One possible encapsulation of NRP-ID in MPLS data plane is proposed in [I-D.li-mpls-enhanced-vpn-vtn-id].

In that case, the NRP ID encapsulated in data packet can be the same value as the NRP ID used in the control protocols, so that the overhead of advertising the mapping relationship between the NRP IDs in the control plane and the corresponding data plane identifiers could be saved.

6. Security Considerations

This document introduces no additional security vulnerabilities to BGP-LS.

The mechanism proposed in this document is subject to the same vulnerabilities as any other protocol that relies on BGP-LS.
7. IANA Considerations

TBD

8. Acknowledgments

The authors would like to thank Shunwan Zhuang and Zhenbin Li for the review and discussion of this document.

9. References

9.1. Normative References

[I-D.ietf-idr-bgp-ls-flex-algo]

[I-D.ietf-idr-bgpls-srv6-ext]

[I-D.ietf-idr-rfc7752bis]

[I-D.ietf-spring-resource-aware-segments]

[I-D.ietf-spring-sr-for-enhanced-vpn]
Dong, J., Bryant, S., Miyasaka, T., Zhu, Y., Qin, F., Li, Z., and F. Clad, "Segment Routing based Virtual Transport Network (VTN) for Enhanced VPN", Work in Progress,
9.2. Informative References

[I-D.dong-6man-enhanced-vpn-vtn-id]

[I-D.dong-lsr-sr-enhanced-vpn]

[I-D.dong-teas-nrp-scalability]

[I-D.ietf-lsr-flex-algo]

[I-D.ietf-lsr-isis-srv6-extensions]


Authors’ Addresses

Jie Dong
Huawei Technologies
Email: jie.dong@huawei.com

Zhibo Hu
Huawei Technologies
Email: huzhibo@huawei.com

Zhenbin Li
Huawei Technologies
Email: lizhenbin@huawei.com

Xiongyan Tang
China Unicom
Abstract

This document describes a BGP based routing solution to establish end-to-end intent-aware paths across a multi-domain service provider transport network. This solution is called BGP Color-Aware Routing (BGP CAR).

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 29, 2022.
Copyright Notice

Copyright (c) 2022 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. Terminology ......................................... 3
   1.2. Illustration ......................................... 5
   1.3. Requirements Language ................................. 7
2. BGP CAR SAFI ........................................... 7
   2.1. Data Model ........................................... 7
   2.2. Extensible encoding .................................. 7
   2.3. BGP CAR Route Origination .............................. 8
   2.4. BGP CAR Route Validation ............................... 8
   2.5. BGP CAR Route Resolution ............................... 8
   2.6. AIGP Metric Computation ............................... 9
   2.7. Path Availability .................................... 9
   2.8. BGP CAR signaling through different color domains .. 10
   2.9. Format and Encoding ................................... 11
       2.9.1. BGP CAR SAFI NLRI Format ........................ 11
       2.9.2. Color-Aware Routes NLRI Type .................... 12
       2.9.3. Local-Color-Mapping (LCM) Extended Community .. 16
   2.10. Error Handling ...................................... 17
3. Service route Automated Steering on Color-Aware path .... 18
4. Intents ............................................... 19
5. (E, C) Subscription and Filtering ........................ 19
   5.1. Illustration ......................................... 19
   5.2. Definition ........................................... 20
6. Scaling ............................................... 20
   6.1. Ultra-Scale Reference Topology ....................... 21
   6.2. Deployment model ..................................... 22
       6.2.1. Flat ............................................. 22
       6.2.2. Hierarchical Design with next-hop-self at ingress domain BR .................................. 23
       6.2.3. Hierarchical Design with Next Hop Unchanged at ingress domain BR .......................... 25
6.3. Scale Analysis .............................................. 26
6.4. Scaling Benefits of the (E, C) BGP Subscription and Filtering .............................................. 28
6.5. Anycast SID .................................................. 28
   6.5.1. Anycast SID for transit inter-domain nodes ........ 28
   6.5.2. Anycast SID for transport color endpoints (e.g., PEs) .... 29
7. Routing Convergence ........................................... 29
8. VPN CAR ....................................................... 29
9. IANA Considerations ........................................... 31
   9.1. BGP CAR NLRI Types Registry .............................. 31
   9.2. BGP CAR NLRI TLV Registry ................................. 31
   9.3. Guidance for Designated Experts ........................... 32
   9.4. BGP Extended Community Registry ........................... 32
10. Acknowledgements ............................................. 32
11. References ................................................... 32
   11.1. Normative References ..................................... 32
   11.2. Informative References ................................... 34
Appendix A. Illustrations of Service Steering .................. 35
   A.1. E2E BGP transport CAR intent realized using IGP FA .... 35
   A.2. E2E BGP transport CAR intent realized using SR Policy .... 37
   A.3. BGP transport CAR intent realized in a section of the network .............................................. 39
   A.4. Transit network domains that do not support CAR ........ 41
Appendix B. Color Mapping Illustrations .......................... 42
   B.1. Single color domain containing network domains with N:N color distribution ................................. 42
   B.2. Single color domain containing network domains with N:M color distribution ................................. 43
   B.3. Multiple color domains ..................................... 43
Authors’ Addresses ................................................. 44

1. Introduction

This document specifies a new BGP SAFI called BGP Color-Aware Routing (BGP CAR). BGP CAR fulfills the transport and VPN problem statement and requirements described in [dskc-bess-bgp-car-problem-statement].

1.1. Terminology

+----------------+-------------------------------------------------------------------------------+
| Intent         | Any combination of the following behaviors: a/ Topology path selection (e.g. minimize metric, avoid resource), b/ NFV service insertion (e.g. service chain steering), c/ per-hop behavior (e.g. 5G slice). |
+----------------+-------------------------------------------------------------------------------+
| Color          | A 32-bit numerical value associated with an intent: e.g. low-cost vs low-delay vs avoiding |
+----------------+-------------------------------------------------------------------------------+
<table>
<thead>
<tr>
<th>Colored Service Route</th>
<th>some resources.</th>
</tr>
</thead>
<tbody>
<tr>
<td>An egress PE E2 colors its BGP VPN route V/v to indicate the intent that it requests for the traffic bound to V/v. The color is encoded as a BGP Color Extended community [I-D.ietf-idr-tunnel-encaps].</td>
<td></td>
</tr>
</tbody>
</table>

| Color-Aware Path to (E2, C) | A routed path to E2 which satisfies the intent associated with color C. Several technologies may provide a Color-Aware Path to (E2, C): SR Policy [I-D.ietf-spring-segment-routing-policy], IGP Flex-Algo [I-D.ietf-lsr-flex-algo], BGP CAR [specified in this document]. |

| Color-Aware Route (E2, C) | A distributed or signaled route that builds a color-aware path to E2 for color C. |
| Service Route Automated Steering on Color-aware path | E1 automatically steers a C-colored service route V/v from E2 onto an (E2, C) path. If several such paths exist, a preference scheme is used to select the best path: E.g. IGP Flex-Algo first then BGP CAR then SR Policy. |

| Color Domain | A set of nodes which share the same Color-to-Intent mapping. This set can be organized in one or several IGP instances or BGP domains. |

| Resolution of a BGP CAR route (E, C) | An inter-domain BGP CAR route (E, C) from N is resolved on an intra-domain color-aware path (N, C) where N is the next-hop of the BGP CAR route. |

| Resolution vs Steering | In this document and consistently with the terminology of the SR Policy document [I-D.ietf-spring-segment-routing-policy], steering is used to describe the mapping of a service route onto a BGP CAR path while the term resolution is preserved for the mapping of an inter-domain BGP CAR route on an intra-domain color-aware path. |

| Service Steering: Service route -> BGP CAR path (or other Color-Aware Routed Paths: e.g., SR Policy) |
| Intra-Domain Resolution: BGP CAR route -> intra-domain color aware path (e.g. SR Policy, IGP Flex-Algo, BGP CAR) |
1.2. Illustration

Here is a brief illustration of the salient properties of the BGP CAR solution.

All the nodes are part of an interdomain network under a single authority and with a consistent color-to-intent mapping:

- C1 is mapped to "low-delay"
  - Flex-Algo FA1 is mapped to "low delay" and hence to C1
- C2 is mapped to "low-delay and avoid resource R"
  - Flex-Algo FA2 is mapped to "low delay and avoid resource R" and hence C2

E1 receives two service routes from E2:

- V/v with BGP Extended-Color community C1
- W/w with BGP Extended-Color community C2

E1 has the following color-aware paths:

- (E2, C1) provided by BGP CAR with the following per-domain support:
  - Domain1: over IGP FA1
  - Domain2: over SR Policy bound to color C1
  - Domain3: over IGP FA1
E1 automatically steers the received service routes as follows:

- V/v via (E2, C1) provided by BGP CAR
- W/w via (E2, C2) provided by SR Policy

Illustrated Properties:

- Leverage of the BGP Color Extended-Community
  * The service routes are colored with widely-used BGP Extended-Color Community
- (E, C) Automated Steering
  * V/v and W/w are automatically steered on the appropriate color-aware path
- Seamless co-existence of BGP CAR and SR Policy
  * V/v is steered on BGP CAR color-aware path
  * W/w is steered on SR Policy color-aware path
- Seamless interworking of BGP CAR and SR Policy
  * V/v is steered on a BGP CAR color-aware path that is itself resolved within domain 2 onto an SR Policy bound to the color of V/v

Other properties:

- MPLS dataplane: with 300k PE’s and 5 colors, the BGP CAR solution ensures that no single node needs to support a dataplane scaling in the order of Remote PE * C. This would otherwise blow the MPLS dataplane.
- Control-Plane: a node should not install a (E, C) path if it does not need it
- Incongruent Color-Intent mapping: the solution supports the signaling of a BGP CAR route across different color domains

The keys to this simplicity are:
o the leverage of the BGP Color Extended-Community to color service routes

o the definition of the automated steering: a C-colored service route V/v from E2 is steered onto a color-aware path (E2, C)

o the definition of the data model of a BGP CAR path: (E, C)

  * consistent with SR Policy data model

o the definition of the recursive resolution of a BGP CAR route: a BGP CAR (E2, C) via N is resolved onto the color-aware path (N, C) which may itself be provided by BGP CAR or via another color-aware routing solution: SR Policy, IGP Flex-Algo.

1.3. 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 CAR SAFI

2.1. Data Model

The BGP CAR data model is:

o NLRI Key: IP Prefix, Color

o NLRI non-key encapsulation data: MPLS label stack, Label index, SRv6 SID list etc.

o BGP Next Hop

o AIGP Metric: accumulates color/intent specific metric across domains

o Local-Color-Mapping Extended-Community (LCM-EC): Optional 32-bit Color value used when a CAR route propagates between different color domains

2.2. Extensible encoding

Extensible encoding is ensured by:
2.3. BGP CAR Route Origination

A BGP CAR route may be originated locally (e.g., loopback) or through redistribution of an (E, C) color-aware path provided by another routing solution: SR Policy, IGP Flex-Algo or BGP-LU [RFC8277].

2.4. BGP CAR Route Validation

A BGP CAR path (E, C) from N with encapsulation T is valid if color-aware path (N, C) exists and T is dataplane available.

A local policy may customize the validation process:

- the color constraint in the first check may be relaxed: instead N is reachable in the default routing table
- the dataplane availability constraint of T may be relaxed
- addition of a performance-measurement verification to ensure that the intent associated with C is met (e.g. delay < bound)

2.5. BGP CAR Route Resolution

A BGP color-aware route (E2, C1) from N is resolved over a color-aware route (N, C1). The color-aware route (N, C1) may be provided recursively by BGP CAR or by other routing solutions: SR Policy, IGP Flex-Algo, BGP-LU.

When multiple resolutions are possible, the default preference should be: IGP Flex-Algo, SR Policy, BGP CAR, BGP LU.

Through local policy, a BGP color-aware route (E2, C1) from N may be resolved over a color-aware route (N, C2): i.e. the local policy maps the resolution of C1 over C2. For example, in a domain where resource R is known to not be present, the inter-domain intent
C1="low delay and avoid R" may be resolved over an intra-domain path of intent C2="low delay".

The color-aware route (N, C1) may have a different dataplane encapsulation than the one of (E2, C1): e.g. a BGP CAR route (E2, C1) with SR-MPLS encapsulation may be transported over an intermediate SRv6 domain.

2.6. AIGP Metric Computation

The Accumulated IGP (AIGP) Attribute is updated as the BGP CAR route propagates across the network.

The value set (or appropriately incremented) in the AIGP TLV corresponds to the metric associated with the underlying intent of the color. For example, when the color is associated with a low-latency path, the metric value is set based on the delay metric.

Information regarding the metric type used by the underlying intra-domain mechanism can also be set.

If BGP CAR routes traverse across a discontinuity in the transport path for a given intent, add a penalty in accumulated IGP metric. The discontinuity is also indicated to upstream nodes via a bit in the AIGP TLV.

AIGP metric computation is recursive.

To avoid continuous IGP metric churn causing end to end BGP CAR churn, an implementation should provide thresholds to trigger AIGP update.

Additional AIGP extensions may be defined to signal state for specific use-cases: MSD along the BGP CAR advertisement, Minimum MTU along the BGP CAR advertisement.

2.7. Path Availability

The (E, C) route inherently provides availability of redundant paths at every hop. For instance, BGP CAR routes originated by two egress ABRs in a domain are advertised as multiple paths to ingress ABRs in the domain, where they become equal-cost or primary-backup paths. A failure of an egress ABR is detected and handled by ingress ABRs locally within the domain for faster convergence, without any necessity to propagate the event to upstream nodes for traffic restoration.
BGP ADD-PATH should be enabled for BGP CAR to signal multiple next hops through a transport RR.

2.8. BGP CAR signaling through different color domains

Let us assume a BGP CAR route (E2, C2) is signaled from B to A; two border routers of respectively domain 2 and domain 1. Let us assume that these two domains do not share the same color-to-intent mapping. Low-delay in domain 2 is color C2 while C1 in domain 1 (C1 <> C2).

The BGP CAR solution seamlessly supports this (rare) scenario while maintaining the separation and independence of the administrative authority in different color domains.

The solution works as follows:

- Within domain 2, the BGP CAR route is (E2, C2) via E2
- B signals to A the BGP CAR route as (E2, C2) via B with Local-Color-Mapping-Extended-Community (LCM-EC) of color C2
- A is aware (classic peering agreement) of the intent-to-color mapping within domain 2 ("low-delay" in domain 2 is C2)
- A maps C2 in LCM-EC to C1 and signals within domain 1 the received BGP CAR route as (E2, C2) via A with LCM-EC(C1)
- The nodes within the receiving domain 1 use the local color encoded in the LCM-EC for next-hop resolution and BGP CAR route installation

Salient properties:

- The NLRI never changes
- E is globally unique, which makes E-C in that order unique
- In the vast majority of the case, the color of the NLRI is used for resolution and steering
- In the rare case of color incongruence, the local color encoded in LCM-EC takes precedence

Further illustrations are provided in Appendix B.
2.9. Format and Encoding

BGP CAR leverages the BGP multi-protocol extensions [RFC4760] and uses the MP_REACH_NLRI and MP_UNREACH_NLRI attributes for route updates by using the SAFI value TBD1 along with AFI 1 for IPv4 prefixes and AFI 2 for IPv6 prefixes.

BGP speakers MUST use BGP Capabilities Advertisement to ensure support for processing of BGP CAR updates. This is done as specified in [RFC4760], by using capability code 1 (multi-protocol BGP), with AFI 1 and 2 (as required) and SAFI TBD1.

The sub-sections below specify the generic encoding of the BGP CAR NLRI followed by the encoding for specific NLRI types introduced in this document.

2.9.1. BGP CAR SAFI NLRI Format

The generic format for the BGP CAR SAFI NLRI is shown below:

```
 0                   1                   2                   3
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  NLRI Length  |  Key Length   |   NLRI Type   |              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+              |
|                  Type-specific Key Fields                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type-specific Non-Key Fields (if applicable)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- NLRI Length: 1 octet field that indicates the length in octets of the NLRI excluding the NLRI Length field itself.

- Key Length: 1 octet field that indicates the length in octets of the NLRI type-specific key fields. Key length MUST be at least 2 less than the NLRI length.

- NLRI Type: 1 octet field that indicates the type of the BGP CAR NLRI.

- Type-Specific Key Fields: Depend on the NLRI type and of length indicated by the Key Length.

- Type-Specific Non-Key Fields: optional and variable depending on the NLRI type. The NLRI encoding allows for encoding of specific
non-key information associated with the route (i.e. the key) as part of the NLRI for efficient packing of BGP updates.

The indication of the key length enables BGP Speakers to determine the key portion of the NLRI and use it along with the NLRI Type field in an opaque manner for handling of unknown or unsupported NLRI types. This can help Route Reflectors (RR) to propagate NLRI types introduced in the future in a transparent manner.

The NLRI encoding allows for encoding of specific non-key information associated with the route (i.e. the key) as part of the NLRI for efficient packing of BGP updates.

The non-key portion of the NLRI MUST be omitted while carrying it within the MP_UNREACH_NLRI when withdrawing the route advertisement.

2.9.2. Color-Aware Routes NLRI Type

The Color-Aware Routes NLRI Type is used for advertisement of color-aware routes and has the following format:

```
+---------------+---------------+-----------------+-------------------+
| NLRI Length   | Key Length    |   NLRI Type   |Prefix Length      |
+---------------+---------------+-----------------+-------------------+
|               |               | IP Prefix (variable) |               //
|               |               | Color (4 octets)                            |
+---------------+---------------+-----------------+-------------------+
```

Followed by optional TLVs encoded as below:

```
+---------------+---------------+-----------------+-------------------+
|     Type      |    Length     |    Value (variable)          //
+---------------+---------------+-----------------+-------------------+
```

where:

- NLRI Length: variable

- Key Length: variable. It indicates the total length comprised of the Prefix Length field, IP Prefix field, and the Color field, as described below. For IPv4 (AFI=1), the minimum length is 5 and maximum length is 9. For IPv6 (AFI=2), the minimum length is 5 and maximum length is 21.

- NLRI Type: 1
Type-Specific Key Fields: as below

* Prefix Length: 1 octet field that carries the length of prefix in bits. Length MUST be less than or equal to 32 for IPv4 (AFI=1) and less than or equal to 128 for IPv6 (AFI=2).

* IP Prefix: IPv4 or IPv6 prefix (based on the AFI). A variable size field that contains the most significant octets of the prefix, i.e., 0 octet for prefix length 0, 1 octet for prefix length 1 to 8, 2 octets for prefix length 9 to 16, 3 octets for prefix length 17 up to 24, 4 octets for prefix length 25 up to 32, and so on. The size of the field MUST be less than or equal to 4 for IPv4 (AFI=1) and less than or equal to 16 for IPv6 (AFI=2).

* Color: 4 octets that contains color value associated with the prefix.

Type-Specific Non-Key Fields: specified in the form of optional TLVs as below:

* Type: 1 octet that contains the type code and flags. It is encoded as shown below:

```
0 1 2 3 4 5 6 7
+--------+-+-+-+
| R | T | Type code |
+--------+-+-+-+
```

where:

+ R: Bit is reserved and MUST be set to 0 and ignored on receive.

+ T: Transitive bit, applicable to speakers that change the BGP CAR next hop

  - T bit set to indicate TLV is transitive. An unrecognized transitive TLV MUST be propagated by a speaker that changes the next hop

  - T bit unset to indicate TLV is non-transitive. An unrecognized non-transitive TLV MUST not be propagated by a speaker that changes next hop

A speaker that does not change next hop should ignore the T-bit and propagate all received TLVs.
+  Type code: Remaining 6 bits contains the type of the TLV.

*  Length: 1 octet field that contains the length of the value portion of the non-key TLV in terms of octets

*  Value: variable length field as indicated by the length field and to be interpreted as per the type field.

The prefix is routable across the administrative domain where BGP transport CAR is deployed. It is possible that the same prefix is originated by multiple BGP CAR speakers in the case of anycast addressing or multi-homing.

The Color is introduced to enable multiple route advertisements for the same prefix. The color is associated with an intent (e.g. low-latency) in originator color-domain.

The following sub-sections specify the non-key TLVs associated with the Color-Aware Routes NLRI type.

2.9.2.1.  Label TLV

The Label TLV is used for advertisement of color-aware routes along with their MPLS labels and has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Followed by one (or more) Labels encoded as below:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Label                 |Rsrv |S|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- Type: Type code is 1. T bit MUST be unset
- Length: variable, MUST be a multiple of 3
- Label Information: multiples of 3 octet fields to convey the MPLS label(s) associated with the advertised color-aware route. It is used for encoding a single label or a stack of labels as per procedures specified in [RFC8277].
When a BGP transport CAR speaker is propagating the route further after setting itself as the nexthop, it allocates a local label for the specific prefix and color combination which it updates in this TLV. It also MUST program a label cross-connect that would result in the label swap operation for the incoming label that it advertises with the label received from its best-path router(s).

2.9.2.2. Label Index TLV

The Label Index TLV is used for advertisement of Segment Routing MPLS (SR-MPLS) Segment Identifier (SID) [RFC8402] information associated with the labeled color-aware routes and 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   |     Flags     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜               |                 Label Index                     ˜
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
˜               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: Type code is 2. T bit MUST be set
- **Length**: 7
- **Reserved**: 1 octet field that MUST be set to 0 and ignored on receipt.
- **Flags**: 2 octet field that maps to the Flags field of the Label-Index TLV of the BGP Prefix SID Attribute [RFC8669].
- **Label Index**: 4 octet field that maps to the Label Index field of the Label-Index TLV of the BGP Prefix SID Attribute [RFC8669].

This TLV provides the equivalent functionality as Label-Index TLV of [RFC8669] for Transport CAR in SR-MPLS deployments. The BGP Prefix SID Attribute SHOULD be omitted from the labeled color-aware routes when the attribute is being used to only convey the Label Index TLV for better BGP packing efficiency.

When a BGP Transport CAR speaker is propagating the route further after setting itself as the nexthop, it allocates a local label for the specific prefix and color combination. When the received update has the Label Index TLV, it SHOULD use that hint to allocate the
local label from the SR Global Block (SRGB) using procedures as specified in [RFC8669].

2.9.2.3. SRv6 SID TLV

BGP Transport CAR can be also used to setup end-to-end color-aware connectivity using Segment Routing over IPv6 (SRv6) [RFC8402]. [I-D.ietf-spring-srv6-network-programming] specifies the SRv6 Endpoint behaviors (e.g. End PSP) which MAY be leveraged for BGP CAR with SRv6. The SRv6 SID TLV is used for advertisement of color-aware routes along with their SRv6 SIDs and has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |    Length     |   SRv6 SID Info (variable)   //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

- **Type**: Type code is 3. T bit MUST be unset
- **Length**: variable, MUST be either less than or equal to 16, or be a multiple of 16
- **SRv6 SID Information**: field of size as indicated by the length that either carries the SRv6 SID(s) for the advertised color-aware route as one of the following:
  - A single 128-bit SRv6 SID or a stack of 128-bit SRv6 SIDs
  - A transposed portion (refer [I-D.ietf-bess-srv6-services]) of the SRv6 SID that MUST be of size in multiples of one octet and less than 16.

The BGP color-aware route update for SRv6 MUST include the BGP Prefix-SID attribute along with the TLV carrying the SRv6 SID information as specified in [I-D.ietf-bess-srv6-services] when using the transposition scheme of encoding for packing efficiency of BGP updates.

2.9.3. Local-Color-Mapping (LCM) Extended Community

This document defines a new BGP Extended Community called "LCM". The LCM is a Transitive Opaque Extended Community with the following encoding:
where:

- Type: 0x3
- Sub-Type: TBD2.
- Reserved: 2 octet of reserved field that MUST be set to zero on transmission and ignored on reception.
- Color: 4-octet field that carries the 32-bit color value.

When a CAR route crosses the originator color domain’s boundary, LCM EC is added. LCM EC conveys the local color mapping for the intent (e.g. low latency) into transit or remote color domains.

The LCM EC MAY be used for filtering of BGP CAR routes and/or for applying routing policies for the intent, when present.

### 2.10. Error Handling

The fault management actions as described in [RFC7606] are applicable for handling of BGP update messages for BGP-CAR.

When the error determined allows for the router to skip the malformed NLRI(s) and continue processing of the rest of the update message, then it MUST handle such malformed NLRIs as ‘Treat-as-withdraw’. In other cases, where the error in the NLRI encoding results in the inability to process the BGP update message, then the router SHOULD handle such malformed NLRIs as ‘AFI/SAFI disable’ when other AFI/SAFI besides BGP-CAR are being advertised over the same session. Alternately, the router MUST perform ‘session reset’ when the session is only being used for BGP-CAR.

Following errors result in ‘AFI/SAFI disable’ or ‘session reset’:

- Minimum NLRI length check error.
- NLRI length conflict with key length.
o Key length encoding errors (such as minimum, maximum and conflict with prefix length).

There can be cases where the NLRI length value is in conflict with the enclosed non-key TLVs, which themselves carry length values. Either the length of a TLV would cause the NLRI length to be exceeded when parsing the TLV, or fewer than 2 bytes remain when beginning to parse the TLV.

In either of these cases, an error condition exists and the "treat-as-withdraw" approach MUST be used (unless some other, more severe error is encountered dictating a stronger approach), and the NLRI Length MUST be relied upon to enable the beginning of the next NLRI field to be located. The above recommendations follow the principle defined in section 4 of [RFC7606].

Type-Specific Non-Key TLV handling

o If multiple instances of same type are encountered, all but the first instance MUST be ignored.

o Type specific length constraints should be verified. The TLV is discarded if there is an error.

o A TLV is not considered malformed because of failing any semantic validation of its Value field.

o Speaker modifying the BGP next-hop MUST recognize at least one of the forwarding information TLV (such as label and SRv6 SID). If it is not able to, such NLRI is considered invalid and not eligible for best path selection.

3. Service route Automated Steering on Color-Aware path

E1 automatically steers a C-colored service route V/v from E2 onto an (E2, C) color-aware path. If several such paths exist, a preference scheme is used to select the best path: E.g. IGP Flex-Algo first then BGP CAR then SR Policy.

This is consistent with the automated service route steering on SR Policy (a routing solution providing color-aware path) defined in [I-D.ietf-spring-segment-routing-policy]. All the steering variations defined in [I-D.ietf-spring-segment-routing-policy] are applicable to BGP CAR color-aware path: on-demand steering, per-destination, per-flow, CO-only. For brevity, in this revision, we refer the reader to the [I-D.ietf-spring-segment-routing-policy] text.
Salient property: Seamless integration of BGP CAR and SR Policy.

Appendix A provides illustrations of service route automated steering.

4. Intents

The widely deployed color-aware path SR Policy solution demonstrates that the following intents can easily be associated with a color:

1. Minimization of a cost metric vs a latency metric
   * Minimization of different metric types, static and dynamic

2. Exclusion/Inclusion of SRLG and/or Link Affinity and/or minimum MTU/number of hops

3. Bandwidth management

4. In the inter-domain context, exclusion/inclusion of entire domains, and border routers

5. Inclusion of one or several virtual network function chains
   * Located in a regional domain and/or core domain, in a DC

6. Localization of the virtual network function chains
   * Some functions may be desired in the regional DC or vice versa

7. Per-Destination and Per-Flow steering

It is straightforward to note that the BGP CAR color-aware alternative supports intents 1, 2, 4 and 7.

Future revisions of this document will analyze the BGP CAR supports for 3, 5 and 6.

5. (E, C) Subscription and Filtering

This section defines an (E, C) BGP subscription model that allows to filter the (E, C) routes learned by a BGP CAR node.

5.1. Illustration
E1-----------------A-------------------B-------------------E2
-- F (E2, C1) --> | --- F (E2, C1) -->
<--- (E2, C1) ---- | --- (E2, C1) ----
<--- (E2, C1) ---- <--- (E2, C1) ----

- BGP CAR route (E2, C1) advertised by E2 is not unconditionally distributed beyond a certain point (e.g., B)
- E1 subscribes to (E2, C1) by advertising a filter route F (E2, C1) to its upstream peer A
- If A has (E2, C1) in its BGP RIB, it will advertise (E2, C1) to E1
- If A does not have (E2, C1), it will advertise F (E2, C1) to its peer B
- B will advertise (E2, C1) to A, which will distribute it to E1

E1 may trigger a subscription for BGP CAR route (E2, C1) as a result of receiving a C1-colored service route V/v from E2, for on-demand steering via (E2, C1).

5.2. Definition

6. Scaling

This section analyses the key scale requirement of [ref:dskc-bess-bgp-car-problem-statement], specifically:

- No intermediate node dataplane should need to scale to (Colors * PEs)
- No node should learn and install a BGP CAR route to (E,C) if it does not install a Colored service route to E

Figure 2 provides an ultra-scale reference topology. Section 6.2 presents three design models to deploy BGP CAR in the reference topology. Section 6.3 analyses the scaling properties of each model. Section 6.4 illustrates the scaling benefits of the (E, C) BGP subscription and filtering.
6.1. Ultra-Scale Reference Topology

The following applies to the reference topology above:

- Independent ISIS/OSPF SR instance in each domain.
- Each domain has Flex Algo 128. Prefix SID for a node is SRGB 168000 plus node number.
- A BGP CAR route (E2, C1) is advertised by egress BRM node 451. The route is sourced locally from redistribution from IGP-FA 128.
- Not shown for simplicity, node 452 will also advertise (E2, C1).
- When a transport RR is used within the domain or across domains, ADD-PATH is enabled to advertise paths from both egress BRs to its clients.
- Egress PE E2 advertises a VPN route RD:V/v with BGP Color extended community C1 that propagates via service RRs to ingress PE E1.
6.2. Deployment model

6.2.1. Flat

1. E1 steers V/v prefix via color-aware path (E2, C1) and VPN label 30030

![Diagram of deployment model]

Figure 3

1. Node 451 advertises BGP CAR route (E2, C1) to 341, from which it goes to 231 then to 121 and finally to E1

2. Each BGP hop allocates local label and programs swap entry in forwarding for (E2, C1)

3. E1 receives BGP CAR route (E2, C1) via 121 with label 168002

1. Let’s assume E1 selects that path
4. E1 resolves BGP CAR route (E2, C1) via 121 on color-aware path (121, C1)

1. Color-aware path (121, C1) is FA128 path to 121 (label 168121)

5. E1’s imposition color-aware label-stack for V/v is thus

1. 30030 <=> V/v
2. 168002 <=> (E2, C1)
3. 168121 <=> (121, C1)

6. Each BGP hop performs swap operation on 168002 bound to color-aware path (E2,C1)

6.2.2. Hierarchical Design with next-hop-self at ingress domain BR

```
(121, C1)
  +-----+  via 121  +-----+
  |       |          |       |
  | 168002|          | 168451|
  +-------+----------+----------
    (E2,C1)    (451,C1)
      /        /        |
     451      121      231
      +-------+        +-------+
        L=168002      L=168451
                        +-----+
                          |E1|
                          +-----+
                        /         /         /       /         /       /         /       /         /         /
                       122      232      342      452
                        +-------+        +-------+        +-------+        +-------+        +-------+
                        Access   Metro   Core   Metro   Access
                        domain 1  domain 2  domain 3  domain 4  domain 5
```

Figure 4: Hierarchical BGP transport CAR, NHS at iBR
1. Node 451 advertises BGP CAR route (451, C1) to 341, from which it goes to 231 and finally to 121

2. Each BGP hop allocates local label and programs swap entry in forwarding for (451, C1)

3. 121 resolves received BGP CAR route (451, C1) via 231 (label 168451) on color-aware path (231, C1)
   1. Color-aware path (231, C1) is FA128 path to 231 (label 168231)

4. 451 advertises BGP CAR route (E2, C1) via 451 to Transport RR T-RR2, which reflects it to T-RR1, which reflects it to 121

5. 121 receives BGP CAR route (E2, C1) via 451 with label 168002
   1. Let’s assume 121 selects that path

6. 121 resolves BGP CAR route (E2, C1) via 451 on color-aware path (451, C1)
   1. Color-aware path (451, C1) is BGP CAR path to 451 (label 168451)

7. 121 imposition of color-aware label stack for (E2, C1) is thus
   1. 168002 <=> (E2, C1)
   2. 168451 <=> (451, C1)
   3. 168231 <=> (231, C1)

8. 121 advertises (E2, C1) to E1 with next hop self (121) and label 168002

9. E1 constructs same imposition color-aware label-stack for V/v via (E2, C1) as in the flat model:
   1. 30030 <=> V/v
   2. 168002 <=> (E2, C1)
   3. 168121 <=> (121, C1)

10. 121 performs swap operation on 168002 with hierarchical color-aware label stack for (E2, C1) via 451 from step 7
11. Nodes 231 and 341 perform swap operation on 168451 bound to color-aware path (451, C1)

12. 451 performs swap operation on 168002 bound to color-aware path (E2, C1)

Note: E1 does not need the BGP CAR (451, C1) route

6.2.3. Hierarchical Design with Next Hop Unchanged at ingress domain BR

![Diagram of hierarchical BGP transport CAR, NHU at iBR]

<table>
<thead>
<tr>
<th>Access domain 1</th>
<th>Metro domain 2</th>
<th>Core domain 3</th>
<th>Metro domain 4</th>
<th>Access domain 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPE 168121</td>
<td>iBRM 168451</td>
<td>iBRC 168451</td>
<td>eBRC 168451</td>
<td>eBRM 168002</td>
</tr>
<tr>
<td>30030</td>
<td>30030</td>
<td>30030</td>
<td>30030</td>
<td>30030</td>
</tr>
</tbody>
</table>

Figure 5: Heirarchical BGP transport CAR, NHU at iBR

1. Nodes 341, 231 and 121 receive and resolve BGP CAR route (451, C1) the same as in the previous model

2. Node 121 allocates local label and programs swap entry in forwarding for (451, C1)

3. 451 advertises BGP CAR route (E2, C1) to Transport RR T-RR2, which reflects it to T-RR1, which reflects it to 121
4. Node 121 advertises (E2, C1) to E1 with next hop as 451 i.e. next-hop unchanged

5. 121 also advertises (451, C1) to E1 with next hop self (121) and label 168451

6. E1 resolves BGP CAR route (451, C1) via 121 on color-aware path (121, C1)
   1. Color-aware path (121, C1) is FA128 path to 121 (label 168121)

7. E1 receives BGP CAR route (E2, C1) via 451 with label 168002
   1. Let’s assume E1 selects that path

8. E1 resolves BGP CAR route (E2, C1) via 451 on color-aware path (451, C1)
   1. Color-aware path (451, C1) is BGP CAR path to 451 (label 168451)

9. E1’s imposition color-aware label-stack for V/v is thus
   1. 30030 <=> V/v
   2. 168002 <=> (E2, C1)
   3. 168451 <=> (451, C1)
   4. 168121 <=> (121, C1)

10. Nodes 121, 231 and 341 perform swap operation on 168451 bound to (451, C1)

11. 451 performs swap operation on 168002 bound to color-aware path (E2, C1)

6.3. Scale Analysis

The following two tables summarize the control-plane and dataplane scale of these three models:
<table>
<thead>
<tr>
<th></th>
<th>E1</th>
<th>121</th>
<th>231</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAT</td>
<td>(E2, C) via (121, C)</td>
<td>(E2, C) via (231, C)</td>
<td>(E2, C) via (341, C)</td>
</tr>
<tr>
<td>H.NHS</td>
<td>(E2, C) via (121, C)</td>
<td>(E2, C) via (451, C)</td>
<td>(451, C) via (231, C)</td>
</tr>
<tr>
<td>H.NHU</td>
<td>(E2, C) via (451, C)</td>
<td>(451, C) via (231, C)</td>
<td>(451, C) via (341, C)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>E1</th>
<th>121</th>
<th>231</th>
</tr>
</thead>
<tbody>
<tr>
<td>FLAT</td>
<td>V -&gt; 30030</td>
<td>168002 -&gt; 168002</td>
<td>168002 -&gt; 168002</td>
</tr>
<tr>
<td></td>
<td>168002</td>
<td>168231</td>
<td>168341</td>
</tr>
<tr>
<td></td>
<td>168121</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H.NHS</td>
<td>V -&gt; 30030</td>
<td>168002 -&gt; 168002</td>
<td>168451 -&gt; 168451</td>
</tr>
<tr>
<td></td>
<td>168002</td>
<td>168451</td>
<td>168341</td>
</tr>
<tr>
<td></td>
<td>168121</td>
<td>168231</td>
<td></td>
</tr>
<tr>
<td>H.NHU</td>
<td>V -&gt; 30030</td>
<td>168451 -&gt; 168451</td>
<td>168451 -&gt; 168451</td>
</tr>
<tr>
<td></td>
<td>168002</td>
<td>168341</td>
<td></td>
</tr>
<tr>
<td></td>
<td>168451</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>168121</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

do The flat model is the simplest design, with a single BGP transport level. It results in the minimum label/SID stack at each BGP hop. However, it significantly increases the scale impact on the core BRs (e.g. 341), whose FIB capacity and even MPLS label space may be exceeded.

* 341’s dataplane scales with (E2, C) where there may be 300k E’s and 5 C’s hence 1.5M entries > 1M MPLS dataplane

o The hierarchical models avoid the need for core BRs to learn routes and install label forwarding entries for (E, C) routes.

* Whether NH self or unchanged at 121, 341’s dataplane scales with (451, C) where there may be thousands of 451’s and 5 C’s hence well under the 1M MPLS dataplane

o The next-hop-self option at ingress BRM (e.g. 121) hides the hierarchical design from the ingress PE, keeping its outgoing label programming as simple as the flat model. However, the ingress BRM requires an additional BGP transport level recursion, which coupled with load-balancing adds dataplane complexity. It
needs to support a swap and push operation. It also needs to install label forwarding entries for the egress PEs that are of interest to its local ingress PEs.

- With the next-hop-unchanged option at ingress BRM (e.g. 121), only an ingress PE needs to learn and install output label entries for egress (E, C) routes. The ingress BRM only installs label forwarding entries for the egress ABR (e.g. 451). However, the ingress PE needs an additional BGP transport level recursion and pushes a BGP VPN label and two BGP transport labels. It may also need to handle load-balancing for the egress ABRs. This is the most complex dataplane option for the ingress PE.

6.4. Scaling Benefits of the (E, C) BGP Subscription and Filtering

The (E, C) subscription scheme from Section 5 provides the following scaling benefits for the models in Section 6.2

- An ingress PE (E1) only learns (E, C) routes that it needs to install into data plane for service route automated steering

- An ingress BRM (121) only learns (E, C) routes that it needs to install into data plane (for Next-Hop-Self), or that it needs to distribute towards it’s ingress PEs (inline RR with Next-Hop-Unchanged)

- An ingress BRM or a transport RR only needs to distribute the necessary subset of (E, C) routes to each client (subscriber); this minimizes their processing load for generating updates

- As a result, withdrawal of (E, C) routes when a remote node fails (E2), may also be faster, aiding better convergence

6.5. Anycast SID

This section describes how Anycast SID complements and improves the scaling designs above.

6.5.1. Anycast SID for transit inter-domain nodes

- Redundant BRs (e.g. two egress BRMs, 451 and 452) advertise BGP CAR routes for a local PE (e.g., E2) with the same SID (based on label-index). Such egress BRMs may be assigned a common Anycast SID, so that the BGP next-hops for these routes will also resolve via a color-aware path to the Anycast SID.

- The use of Anycast SID naturally provides fast local convergence upon failure of an egress BRM node. In addition, it decreases the
recursive resolution and load-balancing complexity at an ingress BRM or PE in the hierarchical designs above.

6.5.2. Anycast SID for transport color endpoints (e.g., PEs)

The common Anycast SID technique may also be used for a redundant pair of PEs that share an identical set of service (VPN) attachments.

- For example, assume a node E2’ paired with E2 above. Both PEs should be configured with the same static label/SID for the services (e.g., per-VRF VPN label/SID), and will advertise associated service routes with the Anycast IP as BGP next-hop.

- This design provides a convergence and recursive resolution benefit on an ingress PE or ABR similar to the egress ABR case above.

7. Routing Convergence

This section will analyze routing convergence.

8. VPN CAR

This section illustrates the extension of BGP CAR to address the VPN CAR requirement stated in Section 3.2 of [dskc-bess-bgp-car-problem-statement].

CE1 -------------- PE1 ------------------------ PE2 -------------- CE2 - V

- BGP CAR is enabled between CE1-PE1 and PE2-CE2
- BGP VPN CAR is enabled between PE1 and PE2
- Provider publishes intent 'low-delay' is mapped to color CP on its inbound peering links
- Within its infrastructure, Provider maps intent 'low-delay' to color CPT
- On CE1 and CE2, intent 'low-delay' is mapped to CC

(V, CC) is a Color-Aware route originated by CE2
1. CE2 sends to PE2 : [(V, CC), Label L1] via CE2 with LCM (CP)
2. PE2 installs in VRF A: [(V, CC), L1] via CE2 which resolves on (CE2, CP)

F
2.a. PE2 allocates VPN Label L2 and programs swap entry for (V, CC)
3. PE2 sends to PE1 : [(RD, V, CC), L2] via PE2 with regular Color Extended Community (CPT)
4. PE1 installs in VRF A: [(V, CC), L2] via (PE2, CPT) steered on (PE2, CPT)
4.a. PE1 allocates Label L3 and programs swap entry for (V, CC)
5. PE1 sends to CE1 : [(V, CC), L3] via PE1 without any LC
6. CE1 installs : [(V, CC), L3] via PE1 which resolves on (PE1, CC)

F
6.a. Label L3 is installed as the imposition label for (V, CC)

VPN CAR distribution for (RD, V, CC) requires a new SAFI that follows same VPN semantics as defined in [RFC4364], the difference being that the advertised routes carry CAR NLRI defined in Section 2.9.2 of this document.

VPN CAR NLRI with RD has the format shown below:

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| NLRI Length | Key Length | NLRI Type | Prefix Length |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Route Distinguisher |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Route Distinguisher |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IP Prefix (variable) // |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Color (4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Followed by optional TLVs encoded as below:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type | Length | Value (variable) // |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

Route Distinguisher: 8 octet field encoded according to [RFC4364]
9. IANA Considerations

IANA is requested to assign SAFI value 83 (BGP CAR) and SAFI value 84 (BGP VPN CAR) from the "SAFI Values" sub-registry under the "Subsequent Address Family Identifiers (SAFI) Parameters" registry with this document as a reference.

9.1. BGP CAR NLRI Types Registry

IANA is requested to create a "BGP CAR NLRI Types" sub-registry under the "Border Gateway Protocol (BGP) Parameters" registry with this document as a reference. The registry is for assignment of the one octet sized code-points for BGP CAR NLRI types and populated with the values shown below:

<table>
<thead>
<tr>
<th>Type</th>
<th>NLRI Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved (not to be used)</td>
<td>[This document]</td>
</tr>
<tr>
<td>1</td>
<td>Color-Aware Routes NLRI</td>
<td>[This document]</td>
</tr>
<tr>
<td>2-255</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

Allocations within the registry are to be made under the "Specification Required" policy as specified in [RFC8126]).

9.2. BGP CAR NLRI TLV Registry

IANA is requested to create a "BGP CAR NLRI TLV Types" sub-registry under the "Border Gateway Protocol (BGP) Parameters" registry with this document as a reference. The registry is for assignment of the one octet sized code-points for BGP-CAR NLRI non-key TLV types and populated with the values shown below:

<table>
<thead>
<tr>
<th>Type</th>
<th>NLRI Type</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved (not to be used)</td>
<td>[This document]</td>
</tr>
<tr>
<td>1</td>
<td>Label TLV</td>
<td>[This document]</td>
</tr>
<tr>
<td>2</td>
<td>Label Index TLV</td>
<td>[This document]</td>
</tr>
<tr>
<td>3</td>
<td>SRv6 SID TLV</td>
<td>[This document]</td>
</tr>
<tr>
<td>4-255</td>
<td>Unassigned</td>
<td></td>
</tr>
</tbody>
</table>

Allocations within the registry are to be made under the "Specification Required" policy as specified in [RFC8126]).
9.3. Guidance for Designated Experts

In all cases of review by the Designated Expert (DE) described here, the DE is expected to ascertain the existence of suitable documentation (a specification) as described in [RFC8126]. The DE is also expected to check the clarity of purpose and use of the requested code points. Additionally, the DE must verify that any request for one of these code points has been made available for review and comment within the IETF: the DE will post the request to the IDR Working Group mailing list (or a successor mailing list designated by the IESG). If the request comes from within the IETF, it should be documented in an Internet-Draft. Lastly, the DE must ensure that any other request for a code point does not conflict with work that is active or already published within the IETF.

9.4. BGP Extended Community Registry

IANA is requested to allocate the sub-type TBD2 for "Local Color Mapping (LCM)" under the "BGP Transitive Opaque Extended Community" registry under the "BGP Extended Community" parameter registry.

10. Acknowledgements

The authors would like to acknowledge the review and inputs from many people.

11. References

11.1. Normative References

[I-D.ietf-bess-srv6-services]

[I-D.ietf-idr-bgp-ipv6-rt-constrain]

[I-D.ietf-idr-tunnel-encaps]
[I-D.ietf-lsr-flex-algo]

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

[I-D.ietf-spring-srv6-network-programming]


11.2. Informative References


Appendix A. Illustrations of Service Steering

The following sub-sections illustrate example scenarios of Colored Service Route Steering over E2E BGP CAR resolving over different intra-domain mechanisms.

The examples use MPLS/SR for the transport data plane. Scenarios specific to other encapsulations will be added in subsequent versions.

A.1. E2E BGP transport CAR intent realized using IGP FA
Use case: Provide end to end intent for service flows.

- With reference to the topology above:
  
  * IGP FA 128 is running in each domain.

  * Egress PE E2 advertises a VPN route RD:V/v colored with (color extended community) C1 to steer traffic to BGP transport CAR (E2, C1). VPN route propagates via service RRs to ingress PE E1.
* BGP CAR route (E2, C1) with next-hop, label-index and label as shown above are advertised through border routers in each domain. When a RR is used in the domain, ADD-PATH is enabled to advertise multiple available paths.

* Local policy on each hop maps intent C1 to resolve CAR route next-hop over IGP FA 128 of the domain. AIGP attribute influences BGP CAR route best path decision as per [RFC7311]. BGP CAR label swap entry is installed that goes over FA 128 LSP to next-hop providing intent in each IGP domain. Update AIGP metric to reflect FA 128 metric to next-hop.

* Ingress PE E1 learns CAR route (E2, C1). It steers colored VPN route RD:v/v into (E2, C1)

  o Important:

  * IGP FA 128 top label provides intent in each domain.

  * BGP CAR label (e.g. 168002) carries end to end intent. Thus stitches intent over intra domain FA 128.

A.2. E2E BGP transport CAR intent realized using SR Policy
Use case: Provide end to end intent for service flows

- With reference to the topology above:
  
  * SR Policy provide intra domain intent.
  
  * Egress PE E2 advertises a VPN route RD:V/v colored with (color extended community) C1 to steer traffic to BGP transport CAR (E2, C1). VPN route propagates via service RRs to ingress PE E1.
  
  * BGP CAR route (E2, C1) with next-hop, label-index and label as shown above are advertised through border routers in each domain. When a RR is used in the domain, ADD-PATH is enabled to advertise multiple available paths.
  
  * Local policy on each hop maps intent C1 to resolve CAR route next-hop over an SR policy(C1, next-hop). BGP CAR label swap entry is installed that goes over SR policy segment list.
* Ingress PE E1 learns CAR route \((E2, C1)\). It steers colored VPN route \(RD:V/v\) into \((E2, C1)\).

- Important:
  - SR policy provides intent in each domain.
  - BGP CAR label (e.g. 168002) carries end to end intent. Thus stitches intent over intra domain SR policies.

A.3. BGP transport CAR intent realized in a section of the network
Use case: Provide intent for service flows only in Core domain.

- With reference to the topology above:
  - IGP FA 128 is only enabled in Core (e.g. WAN network). Access only has base algo 0.
  - Egress PE E2 advertises a VPN route RD:V/v colored with (color extended community) C1 to steer traffic to BGP transport CAR
VPN route propagates via service RRs to ingress PE E1.

* BGP CAR route (E2, C1) with next-hop, label-index and label as shown above are advertised through border routers in each domain. When a RR is used in the domain, ADD-PATH is enabled to advertise multiple available paths.

* Local policy on 231 and 232 maps intent C1 to resolve CAR route next-hop over IGP base algo 0 in right access domain. BGP CAR label swap entry is installed that goes over algo 0 LSP to next-hop. Update AIGP metric to reflect algo 0 metric to next-hop with an additional penalty.

* Local policy on 121 and 122 maps intent C1 to resolve CAR route next-hop learnt from Core domain over IGP FA 128. BGP CAR label swap entry is installed that goes over FA 128 LSP to next-hop providing intent in Core IGP domain.

* Ingress PE E1 learns CAR route (E2, C1). It maps intent C1 to resolve CAR route next-hop over IGP base algo 0. It steers colored VPN route RD:V/v into (E2, C1)

- Important:
  - IGP FA 128 top label provides intent in Core domain.
  - BGP CAR label (e.g. 168002) carries intent from PEs which is realized in core domain

A.4. Transit network domains that do not support CAR

- In a brownfield deployment, color-aware paths between two PEs may need to go through a transit domain that does not support CAR. Example include an MPLS LDP network with IGP best-effort; or a BGP-LU based multi-domain network. MPLS LDP network with best effort IGP can adopt above scheme. Below is the example for BGP LU.

- Reference topology:

  E1 --- BR1 --- BR2 ........ BR3 ---- BR4 --- E2
  \(C1\) \(<-----LU----->\) \(C1\)

  Network between BR2 and BR3 comprises of multiple BGP-LU hops (over IGP-LDP domains).

  * E1, BR1, BR4 and E2 are enabled for BGP CAR, with Ci colors
* BR1 and BR2 are directly connected; BR3 and BR4 are directly connected

- BR1 and BR4 form an over-the-top peering (via RRs as needed) to exchange BGP CAR routes
- BR1 and BR4 also form direct BGP-LU sessions to BR2 and BR3 respectively, to establish labeled paths between each other through the BGP-LU network
- BR1 recursively resolves the BGP CAR next-hop for CAR routes learnt from BR4 via the BGP-LU path to BR4
- BR1 signals the transport discontinuity to E1 via the AIGP TLV, so that E1 can prefer other paths if available
- BR4 does the same in the reverse direction
- Thus, the color-awareness of the routes and hence the paths in the data plane are maintained between E1 and E2, even if the intent is not available within the BGP-LU island
- A similar design can be used for going over network islands of other types

Appendix B. Color Mapping Illustrations

There are a variety of deployment scenarios that arise w.r.t different color mappings in an inter-domain environment. This section attempts to enumerate them and provide clarity into the usage of the color related protocol constructs.

B.1. Single color domain containing network domains with N:N color distribution

- All network domains (ingress, egress and all transit domains) are enabled for the same N colors.
  * A color may of course be realized by different technologies in different domains as described above.
- The N intents are both signaled end-to-end via BGP CAR routes; as well as realized in the data plane.
- Appendix A.1 is an example of this case.
B.2. Single color domain containing network domains with N:M color distribution

- Certain network domains may not be enabled for some of the colors, but may still be required to provide transit.

- When a (E, C) route traverses a domain where color C is not available, the operator may decide to use a different intent of color c that is available in that domain to resolve the next-hop and establish a path through the domain.

  * The next-hop resolution may occur via paths of any intra-domain protocol or even via paths provided by BGP CAR.

  * The next-hop resolution color c may be defined as a local policy at ingress or transit nodes of the domain.

  * It may also be automatically signaled from egress border nodes by attaching a color extended community with value c to the BGP CAR routes.

- Hence, routes of N colors may be resolved via a smaller set of M colored paths in a transit domain, while preserving the original color-awareness end-to-end.

- Any ingress PE that installs a service (VPN) route with a color C, must have C enabled locally to install IP routes to (E, C) and resolve the service route next-hop.

- A degenerate variation of this scenario is where a transit domain does not support any color. Appendix A.3 describes an example of this case.

B.3. Multiple color domains

When the routes are distributed between domains with different color-to-intent mapping schemes, both N:N and N:M cases are possible, although an N:M mapping is more likely to occur.

Reference topology:

```
D1 ------ D2 ------ D3
C1       C2       C3
```

- C1 in D1 maps to C2 in D2 and to C3 in D3

- BGP CAR is enabled in all three domains
The reference topology above is used to elaborate on the design described in Section 2.8

When the route originates in color domain D1 and gets advertised to a different color domain D2, following procedures apply:

- The original intent in the BGP CAR route is preserved; i.e. route is (E, C1)
- A BR of D1 attaches LCM-EC with value C1 when advertising to a BR in D2
- A BR in D2 receiving (E, C1) maps C1 in received LCM-EC to local color, say C2
- Within D2, this LCM-EC value of C2 is used instead of the Color in CAR route NLRI (E, C1). This applies to all procedures described in the earlier section for a single color domain, such as next-hop resolution and installation of route and forwarding entries.
- A colored service route V/v originated in domain D1 with next-hop E and color C1 will also have its color extended-community value re-mapped to C2, typically at a service RR
- On an ingress PE in D2, V/v will resolve via C2
- When a BR in D2 advertises the route to a BR in D3, the same process repeats.

Authors’ Addresses

Dhananjaya Rao
Cisco Systems
USA

Email: dhrao@cisco.com

Swadesh Agrawal
Cisco Systems
USA

Email: swaagraw@cisco.com
Keyur Patel  
Arrcus, Inc  
USA  

Email: keyur@arrcus.com

Haibo Wang  
Huawei Technologies  
China  

Email: rainsword.wang@huawei.com
Interdomain Routing                                      M. Jethanandani
Internet-Draft                                            Kloud Services
Intended status: Standards Track                                K. Patel
Expires: 7 September 2022                                         Arrcus
S. Hares                                               Huawei
J. Haas                                                        Juniper Networks
6 March 2022

BGP YANG Model for Service Provider Networks
draft-ietf-idr-bgp-model-13

Abstract

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

Status of This Memo

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 7 September 2022.

Copyright Notice

Copyright (c) 2022 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
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.

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 ........................... 9
      2.3.1. Local Routing ........................ 11
      2.3.2. Pre updates per-neighbor .............. 11
      2.3.3. Post updates per-neighbor ............. 11
      2.3.4. Pre route advertisements per-neighbor 11
      2.3.5. Post route advertisements per-neighbor 11
3. Relation to other YANG data models ............... 11
4. Security Considerations ........................... 12
5. IANA Considerations ................................ 13
   5.1. URI Registration .......................... 13
   5.2. YANG Module Name Registration ............ 14
6. YANG modules ....................................... 14
7. Structure of the YANG modules .................... 15
   7.1. Main module and submodules for base items 15
   7.2. BGP types ................................ 66
   7.3. BGP policy data ............................ 79
   7.4. RIB modules ................................ 94
8. Contributors ....................................... 124
9. Acknowledgements .................................. 124
10. References ......................................... 124
   10.1. Normative references ..................... 124
   10.2. Informative references ................... 128
Appendix A. Examples ................................ 129
   A.1. Creating BGP Instance ..................... 129
   A.2. Neighbor Address Family Configuration ... 130
   A.3. IPv6 Neighbor Configuration .............. 131
   A.4. VRF Configuration ........................ 132
   A.5. BGP Policy ................................ 134
Appendix B. How to add a new API and Augment a Module 138
Appendix C. How to deviate a module ................ 142
Appendix D. Complete configuration tree diagram ... 142
Appendix E. Complete policy tree diagram .......... 163
Authors’ Addresses .................................. 165
1. Introduction

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

1.1. Goals and approach

The model covers the base BGP features that are deployed across major implementations and the common BGP configurations in use across a number of operator network deployments. In particular, this model attempts to cover BGP features defined in BGP [RFC4271], BGP Communities Attribute [RFC1997], BGP Route Reflection [RFC4456], Multiprotocol Extensions for BGP-4 [RFC4760], Autonomous System Confederations for BGP [RFC5065], BGP Route Flap Damping [RFC2439], Graceful Restart Mechanism for BGP [RFC4724], BGP Prefix Origin Validation [RFC6811], and Advertisement of Multiple Paths in BGP [RFC7911].

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

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

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

* The global BGP instance, and neighbors whose configuration is specified individually, or templated with the use of peer-groups.
* The address families that are supported by peers, and the global configuration which relates to them.

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

* BGP RIB contents.

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

Where possible, naming in the model follows conventions used in available standards documents, and otherwise tries to be self-explanatory with sufficient descriptions of the intended behavior. Similarly, configuration data value constraints and default values, where used, are based on recommendations in current standards documentation, or those commonly used in multiple implementations. Since implementations can vary widely in this respect, this version of the model specifies only a limited set of defaults and ranges with the expectation of being more prescriptive in future versions based on actual operator use.

1.2. Note to RFC Editor

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

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

2022-03-06 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 [RFC9067].
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>

Table 1

2. Model overview

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

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

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

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

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

2.1. BGP protocol configuration

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

```
module: ietf-bgp

augment /rt:routing/rt:control-plane-protocols
   /rt:control-plane-protocol:
     +--rw bgp
     +--rw global!
       |  +--rw as inet:as-number
       |  +--rw identifier? yang:dotted-quad
       |  +--rw distance
       |  |  ...  
       |  +--rw confederation
       |  |  ...  
       |  +--rw graceful-restart (bt:graceful-restart)?
       |  |  ...  
       |  +--rw use-multiple-paths
       |  |  ...  
       |  +--rw route-selection-options
       |  |  ...  
       |  +--rw afi-safis
       |  |  ...  
       |  +--rw apply-policy
       |  |  ...  
       |  +--ro total-paths? uint32
       |  +--ro total-prefixes? uint32
       +--rw neighbors
         |  +--rw neighbor* [remote-address]
         |  |  ...  
         |  +--n established
         |  |  ...
```
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-neighbor or per-peer-group basis, where address-families can be enabled or disabled, and policy associated with the parent entity applied. Within the afi-safi container, generic configuration that applies to all address-families (e.g., whether the AFI-SAFI is enabled) is presented at the top-level, with address-family specific containers made available for options relating to only that AFI-SAFI. Within the current revision of the model a generic set of address-families, and common configuration and state options are included - further work is expected to add additional parameters to this area of the model.

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

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

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

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

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

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

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

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

module: ietf-bgp-policy

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

2.3. BGP RIB overview

The RIB data model represents the BGP RIB contents. The model supports five logical RIBs per address family.
An abridged version of the tree shows the RIB portion of the tree diagram.

module: ietf-bgp

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

The loc-rib is the main BGP routing table for the local routing instance, containing best-path selections for each prefix. The 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. Some of the subtrees and data nodes and their sensitivity/vulnerability are described here.

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

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

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

- The attribute ‘enabled’ in the ‘confederation’ container. This attribute defines whether a local-AS is part of a BGP federation.

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

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. Some of the subtrees and data nodes and their sensitivity/vulnerability are:
The list of neighbors, and their attributes. Allowing a user to read these attributes, in particular the address/port information may allow a malicious user to launch an attack at the particular address/port.

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

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

- The model allows for routes to be cleared using the 'clear' RPC operations, causing the entire RIB table to be cleared.
- The model allows for statistics to be cleared by the 'clear' RPC operation, causing all the individual statistics to be cleared.
- The model also allows for neighbors that have been learnt by the system to be cleared by using the 'clear' RPC operation.

BGP OPSEC [RFC7454] describes several policies that can be used to secure 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

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

5.2. YANG Module Name Registration

This document registers three YANG modules 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

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

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

- ietf-bgp-rib - grouping for representing BGP RIB.
Additionally, modules include:

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

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

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. It references BGP Communities Attribute [RFC1997], Route Refresh Capability for BGP-4 [RFC2918], NOPEER Community for BGP [RFC3765], BGP/MPLS IP Virtual Private Networks (VPNs) [RFC4364], BGP MED Considerations [RFC4451], BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN [RFC4659], Graceful Restart Mechanism for BGP [RFC4724], Multiprotocol Extensions for BGP-4 [RFC4760], Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling [RFC4761], Autonomous System Configuration for BGP [RFC5065], The Generalized TTL Security Mechanism (GTSM) [RFC5082], Bidirectional Forward Detection (BFD) [RFC5880], Bidirectional Forward Detection for IPv4 and IPv6 (Single Hop) [RFC5881], Bidirectional Forwarding Detection (BFD) for Multihop Paths [RFC5883], The TCP Authentication Option [RFC5925], BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs [RFC6514], BGP Support for Four-Octet Autonomous System (AS) Number Space [RFC6793], Advertisement of Multiple Paths in BGP [RFC7911], YANG Key Chain [RFC8177], Carrying Label Information in BGP-4 [RFC8277], A YANG Data Model for Routing Policy [RFC9067], YANG Data Model for Bidirectional Forward Detection [RFC9127], and YANG Model for Transmission Control Protocol (TCP) Configuration [I-D.ietf-tcpm-yang-tcp].

7.1. Main module and submodules for base items

<CODE BEGINS> file "ietf-bgp@2022-03-06.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 (NMBA Version).";
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
```
Internet-Draft               BGP YANG Model                   March 2022

+- [ 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 ]

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', ' SHALL ', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here."

revision 2022-03-06 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network ";
}

/*
 * Identity
 */

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

/*
* Groupings
*/
grouping neighbor-and-peer-group-common {
  description
    "Neighbor and Peer Group configuration that is common."
  
  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;
  }

  container graceful-restart {
    if-feature "bt: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 cleared after all NLRI have been advertised to the peer, and the End-of-RIB (EOR) marker has been cleared.

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

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 rt-pol:apply-policy-group;

/*
 * Containers
 */

augment "/rt:routing/rt:control-plane-protocols/"
+ "rt:control-plane-protocol" {  
  when "derived-from-or-self(rt:type, 'bgp')" {  
    description  
    "This augmentation is valid for a routing protocol instance of BGP.";  
  }  
  description  
  "BGP protocol augmentation of ietf-routing module control-plane-protocol.";  
}  

container bgp {  
  description  
  "Top-level configuration for the BGP router.";  
  container global {  
    presence "Enables global configuration of BGP";  
    description  
    "Global configuration for the BGP router.";  
    leaf as {  
      type inet:as-number;  
      mandatory true;  
      description  
      "Local autonomous system number of the router. Uses the 32-bit as-number type from the model in RFC 6991.";  
    }  
    leaf identifier {  
      type yang:dotted-quad;  
      description  
      "BGP Identifier of the router – an unsigned 32-bit, non-zero integer that should be unique within an AS. The value of the BGP Identifier for a BGP speaker is determined upon startup and is the same for every local interface and BGP peer.";  
      reference  
      "RFC 6286: AS-Wide Unique BGP ID for BGP-4. Section 2.1";  
    }  
    container distance {  
      description  
      "Administrative distances (or preferences) assigned to routes received from different sources (external, and internal).";  
      leaf external {  
        type uint8 {  
          range "1..255";  
        }  
        description  
        "Administrative distances for routes learned from external BGP (eBGP).";  
      }  
      leaf internal {  

type uint8 {
    range "1..255";
} description
    "Administrative distances for routes learned from internal BGP (iBGP).";
}

container confederation {
    description
        "Configuration options specifying parameters when the local router is within an autonomous system which is part of a BGP confederation.";
    leaf enabled {
        type boolean;
        description
            "When this leaf is set to true it indicates that the local-AS is part of a BGP confederation.";
    }
    leaf identifier {
        type inet:as-number;
        description
            "Confederation identifier for the autonomous system.";
    }
    leaf-list member-as {
        type inet:as-number;
        description
            "Remote autonomous systems that are to be treated as part of the local confederation.";
    }
}

container graceful-restart {
    if-feature "bt:graceful-restart";
    description
        "Parameters relating the graceful restart mechanism for BGP.";
    uses graceful-restart-config;
} uses global-group-use-multiple-paths;
uses route-selection-options;
container afi-safis {
    description
        "List of address-families associated with the BGP instance.";
    list afi-safi {
        key "name";
        description
            "AFI,SAFI configuration available for the
neighbor or group.
uses mp-afi-safi-config;
uses state;
container graceful-restart {
  if-feature "bt:graceful-restart";
  description
    "Parameters relating to BGP graceful-restart";
  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 rt-pol:apply-policy-group;
uses state;
}

container neighbors {
  description
    "Configuration for BGP neighbors.";
  list neighbor {
    key "remote-address";
    description
      "List of BGP neighbors configured on the local system,
       uniquely identified by remote IPv[46] address.";
    leaf remote-address {
      type inet:ip-address;
      description
        "The remote IP address of this entry’s BGP peer.";
    }
    leaf local-address {
      type inet:ip-address;
      config false;
      description
        "The local IP address of this entry’s BGP connection.";
    }
    leaf local-port {
      type inet:port-number;
      config false;
      description
        "The local port for the TCP connection between
       the BGP peers.";
    }
  }
}
leaf remote-port {
  type inet:port-number;
  config false;
  description
  "The remote port for the TCP connection between the BGP peers. Note that the objects local-addr, local-port, remote-addr, and remote-port provide the appropriate reference to the standard MIB TCP connection table.";
}
leaf peer-type {
  type bt:peer-type;
  config false;
  description
  "The type of peering session associated with this neighbor.";
  reference
  "RFC 4271: A Border Gateway Protocol 4 (BGP-4) Section 1.1 for iBGP and eBGP.
  RFC 5065: Autonomous System Configuration for Confederation internal and external.";
}
leaf peer-group {
  type leafref {
    path "../../../peer-groups/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 openconfirm or the established state.";
  reference
  "RFC 4271, Section 4.2, 'BGP Identifier'.";
}
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 secure-session-enable {
    type boolean;
    default "false";
    description
    "Does this session need to be secured?";
}

container secure-session {
    when ".../secure-session-enable = 'true'";
    description
    "Container for describing how a particular BGP session is to be secured.";

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

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

description
"Choice of authentication options.";
}

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 neighbor-and-peer-group-common;

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

leaf session-state {
  type enumeration {
    enum idle {
      description
        "Neighbor is down, and in the Idle state of the
         FSM.";
    }
    enum connect {
      description
        "Neighbor is down, and the session is waiting for
         the underlying transport session to be
         established.";
    }
    enum active {
      description
        "Neighbor is down, and the local system is awaiting
         a connection from the remote peer.";
    }
    enum opensent {
      description
        "Neighbor is in the process of being established.
          The local system has sent an OPEN message.";
    }
    enum openconfirm {
      description
        "Neighbor is in the process of being established.
          The local system is awaiting a NOTIFICATION or
          KEEPALIVE message.";
    }
    enum established {
      description
        "Neighbor is up - the BGP session with the peer is
         established.";
    }
  }
}

// notification does not like a non-config statement.
// config false;
description
  "The BGP peer connection state.";
reference
  "RFC 4271, Section 8.1.2.";

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

The BGP session uptime can be computed by clients as the difference between this value and the current time in UTC (assuming the session is in the ESTABLISHED state, per the session-state leaf).";

leaf-list negotiated-capabilities {
  type identityref {
    base bt:bgp-capability;
  }
  config false;
description
  "Negotiated BGP capabilities.";
}

leaf negotiated-hold-time {
  type uint16;
  config false;
description
  "The negotiated hold-time for the BGP session";
}

leaf last-error {
  type binary {
    length "2";
  }
  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."
}
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 bfd {
  if-feature "bt:bfd"
  uses bfd-types:client-cfg-parms;
  description
  "BFD configuration per-neighbor."
}
container statistics {
  description
  "Statistics per neighbor."

  leaf peer-fsm-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
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 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";
    }
}

action clear {
    if-feature "bt:clear-statistics";
    description "Clear statistics action command. Execution of this command should result in all the counters to be cleared and set to 0.";
    input {
        leaf clear-at {
            type yang:date-and-time;
            description "Time when the clear action needs to be executed.";
        }
    }
}
output {
  leaf clear-finished-at {
    type yang:date-and-time;
    description
      "Time when the clear action command completed."
  }
}

notification established {
  leaf remote-address {
    type leafref {
      path "../../neighbor/remote-address";
    }
    description
      "IP address of the neighbor that went into established state."
  }
  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 octet of this two byte OCTET STRING contains the error code, and the second octet 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.";
  }
  description
    "The established event is generated when the BGP FSM enters the established state."
}
notification backward-transition {
  leaf remote-addr {
    type leafref {
      path "../../neighbor/remote-address";
    }
    description
    "IP address of the neighbor that changed its state from
    established state.";
  }
  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.";
  }
  description
  "The backward-transition event is
  generated when the BGP FSM moves from a higher numbered state to a lower numbered state.";
}
action clear {
  if-feature "bt:clear-neighbors";
  description
  "Clear neighbors action.";
  input {
    choice operation {
      default operation-admin;
      description
      "The type of operation for the clear action.";
      case operation-admin {

leaf admin {
  type empty;
  description
  "Closes the Established BGP session with a BGP
  NOTIFICATION message with the Administrative
  Reset error subcode.";
  reference
  "RFC 4486 - Subcodes for BGP Cease Notification
  Message.";
}

case operation-hard {
  leaf hard {
    type empty;
    description
    "Closes the Established BGP session with a BGP
    NOTIFICATION message with the Hard Reset error
    subcode.";
    reference
    "RFC 8538, Section 3 - Notification Message
    Support for BGP Graceful Restart.";
  }
}

case operation-soft {
  leaf soft {
    type empty;
    description
    "Re-sends the current Adj-Rib-Out to this
    neighbor.";
  }
}

case operation-soft-inbound {
  leaf soft-inbound {
    if-feature "bt:route-refresh";
    type empty;
    description
    "Requests the Adj-Rib-In for this neighbor to be
    re-sent using the BGP Route Refresh feature.";
  }
}

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

container peer-groups {
    description
        "Configuration for BGP peer-groups";
    list peer-group {
        key "name";
        description
            "List of BGP peer-groups configured on the local system - uniquely identified by peer-group name";
        leaf name {
            type string;
            description
                "Name of the BGP peer-group";
        }
        leaf secure-session-enable {
            type boolean;
            default "false";
            description
                "Does this session need to be secured?";
        }
        container secure-session {
            when ".../secure-session-enable = ‘true’";
            description
                "Container for describing how a particular BGP session is to be secured.";
            choice option {
                case ao {
                    uses tcp:ao;
                    leaf ao-keychain {
                        type key-chain:key-chain-ref;
                        description
                            "Reference to the key chain that will be used by
this model. Applicable for TCP-AO and TCP-MD5 only;
reference
"RFC 8177: YANG Key Chain.";
}
description
"Uses TCP-AO to secure the session. Parameters for
those are defined as a grouping in the TCP YANG
model.";
reference
"RFC 5925 - The TCP Authentication Option.";
}
case md5 {
uses tcp:md5;
leaf md5-keychain {
  type key-chain:key-chain-ref;
description
"Reference to the key chain that will be used by
this model. Applicable for TCP-AO and TCP-MD5
only";
reference
"RFC 8177: YANG Key Chain.";
}
description
"Uses TCP-MD5 to secure the session. Parameters for
those are defined as a grouping in the TCP YANG
model.";
reference
"RFC 5925: The TCP Authentication Option.";
}
case ipsec {
  leaf sa {
    type string;
description
"Security Association (SA) name.";
}
description
"Currently, the IPsec/IKE YANG model has no
grouping defined that this model can use. When
such a grouping is defined, this model can import
the grouping to add the key parameters
needed to kick of IKE.";
}
description
"Choice of authentication options.";
}
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 neighbor-and-peer-group-common;

container afi-safis {
  description "Per-address-family configuration parameters
              associated with the peer-group.";
  list afi-safi {
    key "name";
    description "AFI, SAFI configuration available for the
                 neighbor or group";
    uses mp-afi-safi-config;
    container graceful-restart {
      if-feature "bt:graceful-restart";
      description "Parameters relating to BGP graceful-restart";
      uses mp-afi-safi-graceful-restart-config;
    }
    uses bgp-neighbor-use-multiple-paths;
    uses mp-all-afi-safi-list-contents;
  }
}

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";
leaf enabled {
  type boolean;
  default "false";
  description
    "Indicates whether BFD is enabled on this interface.";
}
description
  "BFD client configuration.";
reference
  "I-D.ietf-bfd-rfc9127-bis: YANG Data Model for Bidirectional Forward Detection (BFD).";
}description
  "List of interfaces within the routing instance.";
}description
  "Interface specific parameters.";
}uses rib;

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-common@2022-03-06.yang"
submodule ietf-bgp-common {
  yang-version 1.1;
  belongs-to ietf-bgp {
    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-bfd-types {
    prefix bfd-types;
    reference
      "RFC XXXX, YANG Data Model for Bidirectional Forward Detection.";
  }

  Jethanandani, et al. Expires 7 September 2022 [Page 38]
grouping neighbor-group-timers-config {

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

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.”;

revision 2022-03-06 {

description
"Initial Version";

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

"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
    "RFC 4271, Section 4.2.
    RFC 4271, Section 10.";
}
leaf keepalive {
  type uint16 {
    range "0..21845";
  }
}
units "seconds";
When used as a configuration (rw) value, this Time interval (in seconds) for the KeepAlive timer configured for this BGP speaker with this peer. A reasonable maximum value for this timer would be one-third of the configured hold-time.

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

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

The actual time interval for the KEEPALIVE messages is indicated by operational value of keepalive.

reference
[RFC 4271, Section 4.4.
RFC 4271, Section 10.]

leaf min-as-origination-interval {
  type uint16 {
    range "0..max";
  }
  units "seconds";
  description
  "Time interval (in seconds) for the MinASOriginationInterval timer. The suggested value for this timer is 15 seconds."
  reference
  "RFC 4271, 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 peer-as {
    type inet:as-number;
    description
      "AS number of the peer.";
  }
  leaf local-as {
    type inet:as-number;
    description
      "The local autonomous system number that is to be used when
      establishing sessions with the remote peer or peer group, if
      this differs from the global BGP router autonomous system
      number.";
  }
  leaf remove-private-as {
    type bt:remove-private-as-option;
    description
      "When this leaf is specified, remove private AS numbers from
      updates sent to peers.";
  }
  container route-flap-damping {
    if-feature "bt:damping";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enable route flap damping.";
    }
    leaf suppress-above {
      type decimal64 {
        fraction-digits 1;
      }
      default "3.0";
      description
        "This is the value of the instability metric at which
        route suppression takes place. A route is not installed
        in the forwarding information base (FIB), or announced
        even if it is reachable during the period that it is
        suppressed.";
    }
    leaf reuse-above {
      type decimal64 {
        fraction-digits 1;
      }
      default "2.0";
    }
}
description
   "This is the value of the instability metric at which a
   suppressed route becomes unsuppressed if it is reachable
   but currently suppressed. The value assigned to
   reuse-below must be less than suppress-above.";
}
leaf max-flap {
  type decimal64 {
    fraction-digits 1;
  }
  default "16.0";
  description
    "This is the upper limit of the instability metric. This
    value must be greater than the larger of 1 and
    suppress-above.";
}
leaf reach-decay {
  type uint32;
  units "seconds";
  default "300";
  description
    "This value specifies the time desired for the instability
    metric value to reach one-half of its current value when
    the route is reachable. This half-life value determines
    the rate at which the metric value is decayed. A smaller
    half-life value makes a suppressed route reusable sooner
    than a larger value.";
}
leaf unreach-decay {
  type uint32;
  units "seconds";
  default "900";
  description
    "This value acts the same as reach-decay except that it
    specifies the rate at which the instability metric is
    decayed when a route is unreachable. It should have a
    value greater than or equal to reach-decay.";
}
leaf keep-history {
  type uint32;
  units "seconds";
  default "1800";
  description
    "This value specifies the period over which the route
    flapping history is to be maintained for a given route.
    The size of the configuration arrays described below is
    directly affected by this value.";
}
"Routes learned via BGP are subject to weighted route dampening.";
}
leaf-list send-community {
  if-feature "bt:send-communities";
  type identityref {
    base "bt:send-community-feature";
  }
  description
    "When supported, this tells the router to propagate any prefixes that are attached to these community-types.";
}
leaf description {
  type string;
  description
    "An optional textual description (intended primarily for use with a peer or group"
};
}
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 "true";
    description
      "Turns path mtu discovery for BGP TCP sessions on (true) or off (false).";
    reference
      "RFC 1191: Path MTU discovery.";
  }
  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."
}

leaf md5-auth-password {
  type string;
  description
    "Configures an MD5 authentication password for use with
    neighboring devices.";
  reference
    "RFC 2385: Protection of BGP Sessions via the TCP MD5
    Signature Option.";
}

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

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

grouping global-group-use-multiple-paths {
description "Common grouping used for both global and groups which provides configuration and state parameters relating to use of multiple paths";
container use-multiple-paths {
description "Parameters related to the use of multiple paths for the same NLRI";
leaf enabled {
type boolean;
default "false";
description "Whether the use of multiple paths for the same NLRI is enabled for the neighbor. This value is overridden by any more specific configuration value.";
}
}
container ebgp {
  description
    "Multi-Path parameters for eBGP";
  leaf allow-multiple-as {
    type boolean;
    default "false";
    description
      "Allow multi-path to use paths from different neighboring
       ASes. The default is to only consider multiple paths
       from the same neighboring AS.";
  }
  leaf maximum-paths {
    type uint32;
    default "1";
    description
      "Maximum number of parallel paths to consider when using
       BGP multi-path. The default is 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 a shorter length."
}
leaf external-compare-router-id {
  type boolean;
  default "true";
  description
    "When comparing similar routes received from external BGP
    peers, use the router-id as a criterion to select the
    active path."
}
leaf advertise-inactive-routes {
  type boolean;
  default "false";
  description
    "Advertise inactive routes to external peers. The default
    is to only advertise active routes.";
  reference
    "I-D.ietf-idr-best-external: Advertisement of the best
    external route in BGP.";
}
leaf enable-aigp {
  type boolean;
  default "false";
  description
    "Flag to enable sending / receiving accumulated IGP
    attribute in routing updates";
  reference
    "RFC 7311: AIGP Metric Attribute for BGP.";
}
leaf enable-med {
  type boolean;
  default "false";
  description
    "Flag to enable sending/receiving of MED metric attribute

container med-plus-igp {
  leaf enabled {
    type boolean;
    default "false";
    description
      "When enabled allows BGP to use MED and IGP values
       defined below to determine the optimal route."
    reference
      "RFC 4451: BGP MED Considerations.";
  }
  leaf igp-multiplier {
    type uint16;
    default 1;
    description
      "Specifies an IGP cost multiplier.";
    reference
      "RFC 4451: BGP MED Considerations.";
  }
  leaf med-multiplier {
    type uint16;
    default 1;
    description
      "Specifies a MED multiplier.";
    reference
      "RFC 4451: BGP MED Considerations.";
  }
}

description
  "The med-plus-igp option enables BGP to use the sum of
  MED multiplied by a MED multiplier and IGP cost multiplied
  by IGP cost multiplier to select routes when MED is
  required to determine the optimal route.";
}

grouping state {
  description
    "Grouping containing common counters relating to prefixes and
     paths";
  leaf total-paths {
    type uint32;
    config false;
    description
      "Total number of BGP paths (BGP routes) within the context";
  }
  leaf total-prefixes {
type uint32;
config false;
description
"Total number of BGP prefixes (destinations) received within the context";
}
}
}

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-common-multiprotocol@2022-03-06.yang"
submodule ietf-bgp-common-multiprotocol {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix bgp;
  }

  import ietf-bgp-types {
    prefix bt;
  }

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

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

  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 juniper.net).");

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

  Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.
Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject to
the license terms contained in, the Simplified BSD License set
forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
'MAY', and 'OPTIONAL' in this document are to be interpreted as
described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
they appear in all capitals, as shown here.

revision 2022-03-06 {
  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;
    must ". = ./.../.../graceful-restart/enabled";
    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 name {
    type identityref {
      base bt:afi-safi-type;
    }
    description
      "AFI,SAFI";
  }
}
leaf enabled {
  type boolean;
  default "false";
  description
    "This leaf indicates whether this AFI,SAFI is enabled for
    the neighbor or group";
}

grouping mp-all-afi-safi-list-contents {
  description
    "A common grouping used for contents of the list that is used
    for AFI-SAFI entries";
  // import and export policy included for the afi/safi
  uses rt-pol:apply-policy-group;
  container ipv4-unicast {
    when "../name = 'bt:ipv4-unicast'" {
      description
        "Include this container for IPv4 Unicast specific
        configuration";
    }
  }
  container ipv6-unicast {
    when "../name = 'bt:ipv6-unicast'" {
      description
        "Include this container for IPv6 Unicast specific
        configuration";
    }
  }
  container ipv4-labeled-unicast {
    when "../name = 'bt:ipv4-labeled-unicast'" {
      description
        "Include this container for IPv4 Labeled Unicast specific
        configuration";
    }
  }
}
uses mp-all-afi-safi-common;  
// placeholder for IPv4 Labeled Unicast specific config  
// options  
}

container ipv6-labeled-unicast {  
when "./name = ‘bt:ipv6-labeled-unicast’" {  

description  
"Include this container for IPv6 Labeled Unicast specific configuration";  
}

description  
"IPv6 Labeled Unicast configuration options";  
uses mp-all-afi-safi-common;  
// placeholder for IPv6 Labeled Unicast specific config  
// options.  
}

container l3vpn-ipv4-unicast {  
when "./name = ‘bt:l3vpn-ipv4-unicast’" {  

description  
"Include this container for IPv4 Unicast L3VPN specific configuration";  
}

description  
"Unicast IPv4 L3VPN configuration options";  
// include common L3VPN configuration options  
uses mp-l3vpn-ipv4-ipv6-unicast-common;  
// placeholder for IPv4 Unicast L3VPN specific config options.  
}

container l3vpn-ipv6-unicast {  
when "./name = ‘bt:l3vpn-ipv6-unicast’" {  

description  
"Include this container for unicast IPv6 L3VPN specific configuration";  
}

description  
"Unicast IPv6 L3VPN configuration options";  
// include common L3VPN configuration options  
uses mp-l3vpn-ipv4-ipv6-unicast-common;  
// placeholder for IPv6 Unicast L3VPN specific configuration  
// options  
}

container l3vpn-ipv4-multicast {  
when "./name = ‘bt:l3vpn-ipv4-multicast’" {  

description  
"Include this container for multicast IPv6 L3VPN specific configuration";  
}

description  
"Multicast IPv6 L3VPN configuration options";  
// include common L3VPN configuration options  
uses mp-l3vpn-ipv4-ipv6-multicast-common;  
// placeholder for IPv6 Unicast L3VPN specific configuration  
// options  
}
"Multicast IPv4 L3VPN configuration options";
// include common L3VPN multicast options
uses mp-l3vpn-ipv4-ipv6-multicast-common;
// placeholder for IPv4 Multicast L3VPN specific configuration
// options
}
container l3vpn-ipv6-multicast {
  when "../name = 'bt:l3vpn-ipv6-multicast'" {
    description
    "Include this container for multicast IPv6 L3VPN specific configuration";
  }
  description
  "Multicast IPv6 L3VPN configuration options";
  // include common L3VPN multicast options
  uses mp-l3vpn-ipv4-ipv6-multicast-common;
  // placeholder for IPv6 Multicast L3VPN specific configuration
  // options
}
container l2vpn-vpls {
  when "../name = 'bt:l2vpn-vpls'" {
    description
    "Include this container for BGP-signalled VPLS specific configuration";
  }
  description
  "BGP-signalled VPLS configuration options";
  // include common L2VPN options
  uses mp-l2vpn-common;
  // placeholder for BGP-signalled VPLS specific configuration
  // options
}
container l2vpn-evpn {
  when "../name = 'bt:l2vpn-evpn'" {
    description
    "Include this container for BGP EVPN specific configuration";
  }
  description
  "BGP EVPN configuration options";
  // include common L2VPN options
  uses mp-l2vpn-common;
  // placeholder for BGP EVPN specific configuration options
}

// Common groupings across multiple AFI,SAFIs
grouping mp-all-afi-safi-common {
  description
  "Grouping for configuration common to all AFI,SAFI";
  container prefix-limit {
    description
    "Parameters relating to the prefix limit for the AFI-SAFI";
    leaf max-prefixes {
      type uint32;
      description
      "Maximum number of prefixes that will be accepted from the neighbor";
    }
    leaf shutdown-threshold-pct {
      type rt-types:percentage;
      description
      "Threshold on number of prefixes that can be received from a neighbor before generation of warning messages or log entries. Expressed as a percentage of max-prefixes";
    }
    leaf 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.";
    }
  }
}

// include common afi-safi options.
uses mp-all-afi-safi-common;
// configuration options that are specific to IPv[46] unicast
leaf send-default-route {
  type boolean;
  default "false";
  description
  "If set to true, send the default-route to the neighbor(s)";
}

grouping mp-l3vpn-ipv4-ipv6-unicast-common {
  description
  "Common configuration applied across L3VPN for IPv4
and IPv6";
// placeholder -- specific configuration options that are generic
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@2022-03-06.yang"
submodule ietf-bgp-common-structure {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }
    import ietf-routing-policy {
        prefix rt-pol;
        reference
            "RFC ZZZZ, A YANG Data Model for Routing Policy Management";
    }
    import ietf-bgp-types {
        prefix bt;
    }

reference
  "RFC XXXX, BGP YANG Model for Service Provider Network.";
}
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 juniper.net).";

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

Copyright (c) 2021 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject to
the license terms contained in, the Simplified BSD License set
forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents

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

‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as
described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
they appear in all capitals, as shown here."

revision 2022-03-06 {
    description
        "Initial Version";
    reference

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

"RFC XXX, BGP Model for Service Provider Network.";
grouping structure-neighbor-group-route-reflector {
    description "Structural grouping used to include route reflector configuration and state for both BGP neighbors and peer groups";
    container route-reflector {
        description "Route reflector parameters for the BGP peer-group";
        reference "RFC 4456: BGP Route Reflection.";
        leaf cluster-id {
            type bt:rr-cluster-id-type;
            description "Route Reflector cluster id to use when local router is configured as a route reflector. Commonly set at the group level, but allows a different cluster id to be set for each neighbor.";
            reference "RFC 4456: BGP Route Reflection: An Alternative to Full Mesh.";
        }
        leaf 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 in its peer-group to prevent sending of redundant route advertisements.";
        }
        leaf client {
            type boolean;
            default "false";
            description "Configure the neighbor as a route reflector client.";
            reference "RFC 4456: BGP Route Reflection: An Alternative to Full Mesh.";
        }
    }
}

grouping structure-neighbor-group-as-path-options {
    description "Structural grouping used to include AS_PATH manipulation

configuration and state for both BGP neighbors and peer
groups";
container as-path-options {
  description
  "AS_PATH manipulation parameters for the BGP neighbor or
  group";
  leaf allow-own-as {
    type uint8;
    default "0";
    description
    "Specify the number of occurrences of the local BGP
    speaker’s AS that can occur within the AS_PATH before it
    is rejected as looped.";
  }
  leaf replace-peer-as {
    type boolean;
    default "false";
    description
    "Replace occurrences of the peer’s AS in the AS_PATH with
    the local autonomous system number";
  }
}

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

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-neighbor@2022-03-06.yang"
submodule ietf-bgp-neighbor {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix bgp;
  }

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

  // Include the common submodule

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

// meta

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

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

Copyright (c) 2021 IETF Trust and the persons identified as
authors of the code. All rights reserved.

Redistribution and use in source and binary forms, with or
without modification, is permitted pursuant to, and subject to
the license terms contained in, the Simplified BSD License set
forth in Section 4.c of the IETF Trust’s Legal Provisions
Relating to IETF Documents

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
'MAY', and 'OPTIONAL' in this document are to be interpreted as
described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
they appear in all capitals, as shown here.";

revision 2022-03-06 {
    description
        "Initial Version";
    reference
        "RFC XXX, BGP Model for Service Provider Network.";
}

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

grouping bgp-neighbor-counters-message-types-state {
  description "Grouping of BGP message types, included for re-use across counters";
  leaf updates-received {
    type uint64;
    description "Number of BGP UPDATE messages received from this neighbor.";
    reference "RFC 4273: bgpPeerInUpdates.";
  }
  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 "name";
description
  "AFI, SAFI configuration available for the neighbor or
  group";
uses mp-afi-safi-config;
leaf active {
type boolean;
config false;
description
  "This value indicates whether a particular AFI-SAFI has
  been successfully negotiated with the peer. An AFI-SAFI
  may be enabled in the current running configuration, but
  a session restart may be required in order to negotiate
  the new capability.";
}
container prefixes {
  config false;
description
  "Prefix counters for the AFI/SAFI in this BGP session";
leaf received {
type 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 "bt: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-SA
    }
    leaf advertised {
        type boolean;
        config false;
        description
        "This leaf indicates whether the ability to support graceful-restart has been advertised to the peer";
    }
    leaf local-forwarding-state-preserved {
        type boolean;
        config false;
        description
        "This leaf indicates whether the local router has or would advertise the Forwarding State bit in its Graceful Restart capability for this AFI-SA.");
        reference
        "RFC 4724: Graceful Restart Mechanism for BGP."
    }
    leaf forwarding-state-preserved {
        type boolean;
        config false;
        description
        "This leaf indicates whether the neighbor has advertised the Forwarding State bit in its Graceful Restart capability for this AFI-SA.");
        reference
        "RFC 4724: Graceful Restart Mechanism for BGP.";
leaf end-of-rib-received {
  type boolean;
  config false;
  description
    "This leaf indicates whether the neighbor has advertised
    the End-of-RIB marker for this AFI-SAFI."
  reference
    "RFC 4724: Graceful Restart Mechanism for BGP."
}

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

7.2. BGP types

<CODE BEGINS> file "ietf-bgp-types@2022-03-06.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 juniper.net).";

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

  Copyright (c) 2021 IETF Trust and the persons identified as

authors of the code. All rights reserved.

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

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here."

revision 2022-03-06 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

/ *
  * Features.
  */

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.";
}

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

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

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

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

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

/ *
* Identities.
identity bgp-capability {
  description
    "Base identity for a BGP capability";
}

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

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

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

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

identity add-paths {
  if-feature "add-paths";
  base bgp-capability;
  description
    "Advertisement of multiple paths for the same address prefix
     without the new paths implicitly replacing any previous
     ones.";
}
identity afi-safi-type {
  description
    "Base identity type for AFI,SAFI tuples for BGP-4";
  reference
    "RFC4760: Multiprotocol Extentions for BGP-4";
}

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

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

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

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

identity l3vpn-ipv4-unicast {
  base afi-safi-type;
  description
    "Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128)";
  reference
    "RFC 8277: Using BGP to Bind MPLS Labels to Address Prefixes.";
}
identity l3vpn-ipv6-unicast {
  base afi-safi-type;
  description
    "Unicast IPv6 MPLS L3VPN (AFI,SAFI = 2,128)";
  reference
    "RFC 4659: BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN.";
}

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

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

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

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 RFC 1997. These communities must fall within the range 0xFFFF0000 to 0xFFFFFFFF;
reference "RFC 1997: BGP Communities Attribute.";
}

identity no-export {
    base bgp-well-known-std-community;
    description "Do not export NLRI received carrying this community outside the bounds of this autonomous system, or this confederation (if the local autonomous system is a confederation member AS). This community has a value of 0xFFFFFFFF01.";
    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 0xFFFFFFFF02.";
    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 0xFFFFFFFF03.";
    reference "RFC 1997: BGP Communities Attribute.";
}

identity no-peer {
    base bgp-well-known-std-community;
    description "An autonomous system receiving NLRI tagged with this community is advised not to re-advertise the NLRI to external bilateral peer autonomous systems. An AS may also filter received NLRI from bilateral peer sessions when they are tagged with this community value. This community has a value of 0xFFFFFFFF04.";
    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.";
}

identity send-community-feature {
    description

"Base identity to identify send-community feature."
}

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

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

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

/*
 * Typedefs.
*/

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 {
    type union {
        type string {
        }
        type string {
        }
    }
}
description
"Type definition for standard community attributes.";
reference
"RFC 1997 - BGP Communities Attribute";
}
typedef bgp-ext-community-type {
    type union {
        type string {
            // Type 1: 2-octet global and 4-octet local
            // (AS number)        (Integer)
        }
        type string {
            // Type 2: 4-octet global and 2-octet local
            // (ipv4-address)     (integer)
        }
    }
}

type string {
  // route-target with Type 1
  // route-target: (ASN):(local-part)
  // 2 octets global and 4 octets local.
  pattern 'route-target:(6\[0-5\]\[0-5\]\[0-3\]\[0-5\]|' + + '1-5\[0-9\]4\]|1-9\[0-9\]1,4|0-9\]));'
  + '4[0-2]\[0-9\]0-4\[0-9\]0-6\[0-7\]0-2\[0-9\]0-6\]'
  + '[1-3]\[0-9\]9\]|1-9\((0-9\((1,7)\)?0-9\])\[1-9\]));'
}

type string {
  // route-target with Type 2
  // route-target: (IPv4):(local-part)
  // 4 bytes of IP address, and 2 bytes for local.
  pattern 'route-target:'
  + '(([0-9] |[1-9]\[0-9\]|1[0-9]\[0-9\]|2[0-4]\[0-9\]|' + '25[0-5])\.\{3\}\(0-9\)|1-9\[0-9\]1[0-9]\[0-9\]|' + '2[0-4]\[0-9\]25[0-5]):'
  + '6[0-5]\[0-5\]0-3\[0-5\]|1-5\[0-9\]4\]'
  + '[1-9]\[0-9\]1,4|0-9\)])';
}

type string {
  // route-origin with Type 1
  // All 6 octets are open.
  pattern 'route-origin:(6\[0-5\]\[0-5\]\[0-3\]\[0-5\]|' + + '1-5\[0-9\]4\]|1-9\[0-9\]1,4|0-9\]));'
  + '4[0-2]\[0-9\]0-4\[0-9\]0-6\[0-7\]0-2\[0-9\]0-6\]'
  + '[1-3]\[0-9\]9\]|1-9\((0-9\((1,7)\)?0-9\])\[1-9\]));'
}

type string {
  // route-origin with Type 2
  // 4 octets of IP address and two octets of local.
  pattern 'route-origin:'
  + '6[0-5]\[0-5\]0-3\[0-5\]|1-5\[0-9\]4\]'
  + '[1-9]\[0-9\]1,4|0-9\]));'
}

description
"Type definition for extended community attributes";
reference
"RFC 4360 - BGP Extended Communities Attribute";
typedef bgp-community-regexp-type {
    type string;
    description
        "Type definition for communities specified as regular
         expression patterns";
}

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

typedef peer-type {
    type enumeration {
        enum internal {
            description
        }
    }
}

"Internal (IBGP) peer";
}
enum external {
  description
  "External (EBGP) peer";
}
enum confederation-internal {
  description
  "Confederation Internal (IBGP) peer.";
}
enum confederation-external {
  description
  "Confederation External (EBGP) peer.";
}

description
"Labels a peer or peer group as explicitly internal, external, or the related confederation type.";
reference
"RFC 4271 - A Border Gateway Protocol 4 (BGP-4), Sec 1.1. RFC 5065, Autonomous System Configuration for BGP.";

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 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";
}

7.3. BGP policy data

<CODE BEGINS> file "ietf-bgp-policy@2022-03-06.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 rt-pol;
    }
    import ietf-bgp-types {
        prefix bt;
    }
    import ietf-routing-types {
        prefix rt-types;
    }

    organization
        "IETF IDR Working Group";
    contact
This module contains data definitions for BGP routing policy. It augments the base routing-policy module with BGP-specific options for conditions and actions.

Copyright (c) 2022 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

revision 2022-03-06 {
    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
                "WG Web: <http://datatracker.ietf.org/wg/idr>
            WG List: <idr@ietf.org>
            Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                     Keyur Patel (keyur at arrcus.com),
                     Susan Hares (shares at ndzh.com),
                     Jeffrey Haas (jhaas at juniper.net)."
        }
    }
}
"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]{3}|[0-9]{1,9}|4[0-1][0-9]{1,8}|428[0-9]{1,7}|429[0-3][0-9]{1,9}|42948[0-9]{1,5}|42949[0-5][0-9]{1,6}|429496[0-9]{1,7}|429497[0-9]{1,8}|42949728[0-9]{1,9}|42949729[0-5])$';
      }
    }
  }
  type enumeration {
    enum igp { 
      description
      }
    }
  }
  description
  "Type definition for specifying MED in policy actions.";
}
"Set the MED value to the IGP cost toward the next hop for the route."

}  

enum med-plus-igp {
    description
    "Before comparing MED values for path selection, adds to the MED the cost of the IGP route to the BGP next-hop destination.

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

    Useful when the downstream AS requires the complete cost of a certain route that is received across multiple ASs.";
}

// Identities

// augment statements

augment "/rt-pol:routing-policy/rt-pol:defined-sets" {
    description
    "Adds BGP defined sets container to routing policy model.";
    container bgp-defined-sets {
        description
        "BGP-related set definitions for policy match conditions.";
        container community-sets {
            description
            "Enclosing container for list of defined BGP community sets.";
            list community-set {
                key "name";
                description
                "List of defined BGP community sets.";
            }
        }
    }
}
leaf name {
  type string;
  description
      "Name / label of the community set -- this is used to
       reference the set in match conditions.";
}
leaf-list member {
  type union {
    type rt-types:route-target;
    type bt:bgp-community-regexp-type;
    type bt:bgp-well-known-community-type;
  }
  description
      "Members of the community set";
}
}

container ext-community-sets {
  description
      "Enclosing container for list of extended BGP community
       sets";
  list ext-community-set {
    key "name";
    description
      "List of defined extended BGP community sets";
    leaf name {
      type string;
      description
          "Name / label of the extended community set -- this is
           used to reference the set in match conditions";
    }
    leaf-list member {
      type union {
        type rt-types:route-target;
        type bt:bgp-community-regexp-type;
      }
      description
          "Members of the extended community set.";
    }
  }
}

container large-community-sets {
  description
      "Enclosing container for list of large BGP community
       sets";
  list large-community-set {

}
key "name";

description
"List of defined large BGP community sets";
leaf name {
    type string;
    description
    "Name / label of the large community set -- this is used to reference
    the set in match conditions";
}
leaf-list member {
    type union {
        type bt:bgp-large-community-type;
        type bt:bgp-community-regexp-type;
    }
    description
    "Members of the large 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 {
            type string;
            description
            "AS path regular expression -- list of ASes in the set.";
        }
    }
}

container next-hop-sets {
    description
    "Definition of a list of IPv4 or IPv6 next-hops which can be matched
    in a routing policy.";
    list next-hop-set {

key "name"
  description
    "List of defined next-hop sets for use in policies.";

leaf name {
  type string;
  description
    "Name of the next-hop set.";
}

leaf-list next-hop {
  type bgp-next-hop-type;
  description
    "List of IP addresses in the next-hop set.";
}

augment "/rt-pol:routing-policy/rt-pol:policy-definitions/"
  + "rt-pol:policy-definition/rt-pol:statements/"
  + "rt-pol:statement/rt-pol:conditions" {
  description
    "BGP policy conditions added to routing policy module.";

container bgp-conditions {
  description
    "Top-level container for BGP specific policy conditions.";

  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-eq {
    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;
    description
        "Condition to check if the local prefer attribute is equal to
        the specified value.";
}

leaf-list neighbor-eq {
    type inet:ip-address;
    description
        "List of neighbor addresses to check for in the ingress
        direction.";
}

leaf route-type {
    type enumeration {
        enum internal {
            description
                "route type is internal.";
        }
        enum external {
            description
                "route type is external.";
        }
    }
    description
        "Condition to check the route type in the route update.";
}

container community-count {
    description
        "Value and comparison operations for conditions based on
        the number of communities in the route update.";
}

leaf community-count {
    type uint32;
    description
        "Value for the number of communities in the route"
update.
}

choice operation {
    case eq {
        leaf eq {
            type empty;
            description
                "Check to see if the value is equal.";
        }
    }

    case lt-or-eq {
        leaf lt-or-eq {
            type empty;
            description
                "Check to see if the value is less than or equal.";
        }
    }

    case gt-or-eq {
        leaf gt-or-eq {
            type empty;
            description
                "Check to see if the value is greater than or equal.";
        }
    }

description
    "Choice of operations on the value of community-count.";
}

container as-path-length {
    description
        "Value and comparison operations for conditions based on
         the length of the AS path in the route update.

         The as-path-length SHALL be calculated and SHALL follow
         RFC 4271 rules.";
    reference
        "RFC 4271: BGP-4.";

    leaf as-path-length {
        type uint32;
        description
            "Value of the AS path length in the route update.";
    }
}
choice operation {
  case eq {
    leaf eq {
      type empty;
      description
        "Check to see if the value is equal.";
    }
  }
  case lt-or-eq {
    leaf lt-or-eq {
      type empty;
      description
        "Check to see if the value is less than or equal.";
    }
  }
  case gt-or-eq {
    leaf gt-or-eq {
      type empty;
      description
        "Check to see if the value is greater than or equal.";
    }
  }
}

description
  "Choice of operations on the value of as-path-len.";
}
}

container match-community-set {
  description
    "Top-level container for match conditions on communities.
    Match a referenced community-set according to the logic
    defined in the match-set-options leaf."
  leaf community-set {
    type leafref {
      path ":/rt-pol:routing-policy/rt-pol:defined-sets/"
        + "bgp-defined-sets/community-sets/"
        + "community-set/name";
    }
    description
      "References a defined community set.";
  }
  uses rt-pol:match-set-options-group;
}

container match-ext-community-set {

description
  "Match a referenced extended community-set according to the
  logic defined in the match-set-options leaf.";
leaf ext-community-set {
  type leafref {
    path "/rt-pol:routing-policy/rt-pol:defined-sets/
       + "bgp-defined-sets/ext-community-sets/
       + "ext-community-set/name";
  }
  description
    "References a defined extended community set.";
} uses rt-pol:match-set-options-group;
}

container match-large-community-set {
  description
    "Match a referenced large community-set according to the
    logic defined in the match-set-options leaf.";
  leaf ext-community-set {
    type leafref {
      path "/rt-pol:routing-policy/rt-pol:defined-sets/
         + "bgp-defined-sets/large-community-sets/
         + "large-community-set/name";
    }
    description
      "References a defined large community set.";
  }
  uses rt-pol:match-set-options-group;
}

container match-as-path-set {
  description
    "Match a referenced as-path set according to the logic
    defined in the match-set-options leaf.";
  leaf as-path-set {
    type leafref {
      path "/rt-pol:routing-policy/rt-pol:defined-sets/
         + "bgp-defined-sets/as-path-sets/
         + "as-path-set/name";
    }
    description
      "References a defined AS path set";
  }
  uses rt-pol:match-set-options-group;
}

container match-next-hop-set {

description
"Match a referenced next-hop set according to the logic
defined in the match-set-options leaf."
leaf next-hop-set {
    type leafref {
        path "/rt-pol:routing-policy/rt-pol:defined-sets/
            bgp-defined-sets/next-hop-sets/
            next-hop-set/name";
    }
    description
    "Reference a defined next-hop set."
}
uses rt-pol:match-set-options-group;
}
}
}
}

augment "/rt-pol:routing-policy/rt-pol:policy-definitions/
    rt-pol:policy-definition/rt-pol:statements/
    rt-pol:statement/rt-pol:actions" {
    description
    "BGP policy actions added to routing policy module."
    container bgp-actions {
        description
        "Top-level container for BGP-specific actions"
        leaf set-route-origin {
            type bt:bgp-origin-attr-type;
            description
            "Set the origin attribute to the specified value"
        }
        leaf set-local-pref {
            type uint32;
            description
            "Set the local pref attribute on the route."
        }
        leaf set-next-hop {
            type bgp-next-hop-type;
            description
            "Set the next-hop attribute in the route."
        }
        leaf set-med {
            type bgp-set-med-type;
            description
            "Set the med metric attribute in the route."
        }
        container set-as-path-prepend {
            description
            "Action to prepend local AS number to the AS-path a
specified number of times);

leaf repeat-n {
  type uint8 {
    range "1..max";
  }
  description
  "Number of times to prepend the 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).";

  leaf options {
    type bgp-set-community-option-type;
    description
    "Options for modifying the community attribute with the specified values. These options apply to both methods of setting the community attribute.";
  }

  choice method {
    description
    "Indicates the method used to specify the extended communities for the set-community action";
    case inline {
      leaf-list communities {
        type union {
          type bt:bgp-std-community-type;
          type bt:bgp-well-known-community-type;
        }
        description
        "Set the community values for the update inline with a list.";
      }
    }

    case reference {
      leaf community-set-ref {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:defined-sets/"
        }
      }
    }
  }
}
container set-ext-community {
  description
  "Action to set the extended community attributes of the route, along with options to modify how the community is modified. Extended communities may be set using an inline list OR a reference to an existing defined set (but not both).";

  leaf options {
    type bgp-set-community-option-type;
    description
    "Options for modifying the community attribute with the specified values. These options apply to both methods of setting the community attribute.";
  }

  choice method {
    description
    "Indicates the method used to specify the extended communities for the set-ext-community action";
  case inline {
    leaf-list communities {
      type rt-types:route-target;
      description
      "Set the extended community values for the update inline with a list.";
    }
  }
  case reference {
    leaf ext-community-set-ref {
      type leafref {
        path "/rt-pol:routing-policy/rt-pol:defined-sets/"
        + "bgp-defined-sets/ext-community-sets/"
        + "ext-community-set/name";
      }
      description
      "References a defined extended community set by name.";
    }
  }
}
container set-large-community {
  description
  "Action to set the large community attributes of the route, along with options to modify how the community is modified. Large communities may be set using an inline list OR a reference to an existing defined set (but not both).";

  leaf options {
    type bgp-set-community-option-type;
    description
    "Options for modifying the community attribute with the specified values. These options apply to both methods of setting the community attribute.";
  }

  choice method {
    description
    "Indicates the method used to specify the large communities for the set-large-community action";

    case inline {
      leaf-list communities {
        type bt:bgp-large-community-type;
        description
        "Set the large community values for the update inline with a list.";
      }
    }

    case reference {
      leaf large-community-set-ref {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:defined-sets/"
          + "bgp-defined-sets/large-community-sets/"
          + "large-community-set/name";
        }
        description
        "References a defined extended community set by name.";
      }
    }
  }
}
7.4. RIB modules

<CODE BEGINS> file "ietf-bgp-rib@2022-03-06.yang"
submodule ietf-bgp-rib {  
yang-version 1.1;
belongs-to ietf-bgp {  
  prefix br;
}

/*
 * Import and Include
 */

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

organization
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.

Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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

The key words ‘MUST’, ‘MUST NOT’, ‘REQUIRED’, ‘SHALL’, ‘SHALL NOT’, ‘SHOULD’, ‘SHOULD NOT’, ‘RECOMMENDED’, ‘NOT RECOMMENDED’, ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here."

revision 2022-03-06 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network.";
}

grouping attr-set-attributes {
  description
    "A grouping for all attribute set parameters.";

carrier attributes {
  description
    "A container for attribute set parameters.";

  leaf origin {
    type bt:bgp-origin-attr-type;
    description
      "BGP attribute defining the origin of the path information.";
  }

  leaf atomic-aggregate {
    type boolean;
    description
      "BGP attribute indicating that the prefix is an atomic aggregate; i.e., the peer selected is a less specific route without selecting a more specific route that is subsumed by it.";
    reference
      "RFC 4271: 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 5.1.3.";
}
leaf link-local-next-hop {
  type inet:ipv6-address;
description
  "When both a global and a link-local next-hop are sent
  when following RFC 2545 procedures, this leaf contains
  the link-local next-hop."
  reference
  "RFC 2545: Use of BGP-4 Multiprotocol Extensions for IPv6
  Inter-Domain Routing";
}
leaf med {
  type uint32;
description
  "BGP multi-exit discriminator attribute used in the BGP
  route selection process."
  reference
  "RFC 4271: 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
leaf aigp-metric {
    type uint64;
    description "BGP path attribute representing the accumulated IGP metric for the path";
    reference "RFC 7311 - The Accumulated IGP Metric Attribute for BGP";
}

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.
RFC 6793 - BGP Support for Four-octet AS Number Space.";
    leaf as {
        type inet:as-number;
        description "AS number of the autonomous system that performed the aggregation.";
    }
    leaf address {
        type inet:ipv4-address;
        description "IP address of the router that performed the aggregation.";
    }
}

container aggregator4 {
    config false;
    description "BGP attribute indicating the prefix has been aggregated by the specified AS and router. This value is populated with the received or sent attribute in Adj-RIB-In or Adj-RIB-Out, respectively. It should not be populated in Loc-RIB since the Loc-RIB is expected to store the effective AGGREGATOR in the aggregator/as leaf regardless of being 4-octet or 2-octet.";
    reference "RFC 4271: Section 5.1.7.";
    leaf as4 {
        "RFC 4456 - BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)";
    }
}

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.";
  reference
  "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)
  RFC 6793 - BGP Support for Four-octet AS Number Space
  RFC 5065 - Autonomous System Confederations for BGP";
  list segment {
    config false;
    uses bgp-as-path-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 {
  config false;
  uses bgp-as-path-attr;
  description
    "List of AS PATH segments";
}
}
}
grouping attr-set {
  description
    "A grouping for all path attributes.";

list attr-set {
  key "index";
  description
    "List of path attributes that may be in use by multiple
    routes in the table";
  leaf index {
    type uint64;
    description
      "System generated index for each attribute set. The
      index is used to reference an attribute set from a
      specific path. Multiple paths may reference the same
      attribute set.";
  }
  uses attr-set-attributes;
}
}
grouping attr-sets {
  description
    "A grouping for all sets of path attributes.";

container attr-sets {
  description
    "Enclosing container for the list of path attribute sets";
  uses attr-set;
}
}
grouping ext-community-attributes {
  description
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."
  reference
    "RFC 4360 - BGP Extended Communities Attribute";
}

leaf-list large-community {
  type bt:bgp-large-community-type;
  description
    "List of BGP large community attributes."
  reference
    "RFC 8092: BGP Large Communities Attribute.";
}

container rib {
  description
    "Grouping for rib.";
  container rib {
    config false;
    uses attr-sets;
    container communities {
      description
        "Enclosing container for the list of community attribute sets."
      list community {
        key "index";
        config false;
        description
          "List of path attributes that may be in use by multiple routes in the table."
        leaf index {
          type uint64;
          description
            "System generated index for each attribute set. The index is used to reference an attribute set from a
specific path. Multiple paths may reference the same attribute set.
}
}
container ext-communities {
  description
      "Enclosing container for the list of extended community attribute sets.";
  list ext-community {
    key "index";
    config false;
    description
        "List of path attributes that may be in use by multiple routes in the table.";
    leaf index {
      type uint64;
      description
        "System generated index for each attribute set. The index is used to reference an attribute set from a specific path. Multiple paths may reference the same attribute set.";
    }
    uses ext-community-attributes;
  }
}
container large-communities {
  description
      "Enclosing container for the list of large community attribute sets.";
  list large-community {
    key "index";
    config false;
    description
        "List of path attributes that may be in use by multiple routes in the table.";
    leaf index {
      type uint64;
      description
        "System generated index for each attribute set. The index is used to reference an attribute set from a specific path. Multiple paths may reference the same attribute set.";
    }
    uses large-community-attributes;
  }
}
container afi-safis {
  config false;
  description
    "Enclosing container for address family list.";
  list afi-safi {
    key "name";
    description
      "List of afi-safi types.";
    leaf name {
      type identityref {
        base bt:afi-safi-type;
      }
      description
        "AFI,SAFI name.";
    }
  }
  container ipv4-unicast {
    when "../name = 'bt:ipv4-unicast'" {
      description
        "Include this container for IPv4 unicast RIB.";
    }
  }
  description
    "Routing tables for IPv4 unicast -- active when the
     afi-safi name is ipv4-unicast.";

counter loc-rib {
  config false;
  description
    "Container for the IPv4 BGP LOC-RIB data.";
  container routes {
    description
      "Enclosing container for list of routes in the
       routing table.";
    list route {
      key "prefix origin path-id";
      description
        "List of routes in the table, keyed by the route
         prefix, the route origin, and path-id. The route
         origin can be either the neighbor address from
         which the route was learned, or the source
         protocol that injected the route. The path-id
         distinguishes routes for the same prefix
         received from a neighbor (e.g., if add-paths is
         enabled).";
      leaf prefix {
        type inet:ipv4-prefix;
        description
          "The IPv4 prefix corresponding to the route.";
      }
    }
  }
}
uses bgp-loc-rib-common-keys;
uses bgp-loc-rib-common-attr-refs;
uses bgp-common-route-annotations-state;
uses bgp-unknown-attr-top;
uses rib-ext-route-annotations;
}
}
}

container neighbors {
  config false;
  description
      "Enclosing container for neighbor list.";
  list neighbor {
    key "neighbor-address";
    description
      "List of neighbors (peers) of the local BGP
       speaker.";
    leaf neighbor-address {
      type inet:ip-address;
      description
        "IP address of the BGP neighbor or peer.";
    }
  }
  container adj-rib-in-pre {
    description
      "Per-neighbor table containing the 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;
    uses clear-routes {
      description
        "Clears the adj-rib-in state for the containing
         neighbor. Subsequently, implementations might
         issue a 'route refresh' if 'route refresh' has
         been negotiated, or reset the session.";
    }
  }
  container adj-rib-in-post {
    description
      "Per-neighbor table containing the paths received
       from the neighbor that are eligible for
       best-path selection after local input policy
       rules have been applied.";
    uses ipv4-adj-rib-in-post;
    uses clear-routes {
      description

"Clears the adj-rib-in state for the containing neighbor. Subsequently, implementations might issue a 'route refresh' if 'route refresh' has been negotiated, or reset the session.";

container adj-rib-out-pre {
 description
 "Per-neighbor table containing paths eligible for sending (advertising) to the neighbor before output policy rules have been applied.";
 uses ipv4-adj-rib-common;
 uses clear-routes {
 description
 "Clears the adj-rib-out state for the containing neighbor. Subsequently, neighbors will announce BGP updates to resynchronize these routes.";
 }
}

container adj-rib-out-post {
 description
 "Per-neighbor table containing paths eligible for sending (advertising) to the neighbor after output policy rules have been applied.";
 uses ipv4-adj-rib-common;
 uses clear-routes {
 description
 "Clears the adj-rib-out state for the containing neighbor. Subsequently, neighbors will announce BGP updates to resynchronize these routes.";
 }
}

container ipv6-unicast {
 when ".../name = 'bt:ipv6-unicast'" {
 description
 "Include this container for IPv6 unicast RIB.";
 }
 description
 "Routing tables for IPv6 unicast -- active when the afi-safi name is ipv6-unicast.";
}

container loc-rib {
config false;
description
"Container for the IPv6 BGP LOC-RIB data."
container routes {
    description
    "Enclosing container for list of routes in the
    routing table."
    list route {
        key "prefix origin path-id";
        description
        "List of routes in the table, keyed by the route
        prefix, the route origin, and path-id. The route
        origin can be either the neighbor address from
        which the route was learned, or the source
        protocol that injected the route. The path-id
        distinguishes routes for the same prefix
        received from a neighbor (e.g., if add-paths is
        enabled)."
        leaf prefix {
            type inet:ipv6-prefix;
            description
            "The IPv6 prefix corresponding to the route."
        }
        uses bgp-loc-rib-common-keys;
        uses bgp-loc-rib-common-attr-refs;
        uses bgp-common-route-annotations-state;
        uses bgp-unknown-attr-top;
        uses rib-ext-route-annotations;
    }
}
}

container neighbors {
    config false;
    description
    "Enclosing container for neighbor list."
    list neighbor {
        key "neighbor-address";
        description
        "List of neighbors (peers) of the local BGP
        speaker."
        leaf neighbor-address {
            type inet:ip-address;
            description
            "IP address of the BGP neighbor or peer."
        }
    }
    container adj-rib-in-pre {
        description
        "List of routes in the table, keyed by the route
        prefix, the route origin, and path-id. The route
        origin can be either the neighbor address from
        which the route was learned, or the source
        protocol that injected the route. The path-id
        distinguishes routes for the same prefix
        received from a neighbor (e.g., if add-paths is
        enabled)."
        leaf prefix {
            type inet:ipv6-prefix;
            description
            "The IPv6 prefix corresponding to the route."
        }
        uses bgp-loc-rib-common-keys;
        uses bgp-loc-rib-common-attr-refs;
        uses bgp-common-route-annotations-state;
        uses bgp-unknown-attr-top;
        uses rib-ext-route-annotations;
    }
}
"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;
uses clear-routes {
    description
    "Clears the adj-rib-in state for the containing neighbor. Subsequently, implementations might issue a 'route refresh' if 'route refresh' has been negotiated, or reset the session. ";
}
}

container adj-rib-in-post {
    description
    "Per-neighbor table containing the paths received from the neighbor that are eligible for best-path selection after local input policy rules have been applied."
uses ipv6-adj-rib-in-post;
uses clear-routes {
    description
    "Clears the adj-rib-in state for the containing neighbor. Subsequently, implementations might issue a 'route refresh' if 'route refresh' has been negotiated, or reset the session. ";
}
}

container adj-rib-out-pre {
    description
    "Per-neighbor table containing paths eligible for sending (advertising) to the neighbor before output policy rules have been applied."
uses ipv6-adj-rib-common;
uses clear-routes {
    description
    "Clears the adj-rib-out state for the containing neighbor. Subsequently, neighbors will announce BGP updates to resynchronize these routes."
}
}

container adj-rib-out-post {
    description
    "Per-neighbor table containing paths eligible for sending (advertising) to the neighbor after output policy rules have been applied.";
uses ipv6-adj-rib-common;
uses clear-routes {
  description
  "Clears the adj-rib-out state for the containing neighbor. Subsequently, neighbors will announce BGP updates to resynchronize these routes.";
}
}
}

description
  "Top level container for BGP RIB.";
}
}
}

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-rib-types@2022-03-06.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 juniper.net)."

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

  Copyright (c) 2021 IETF Trust and the persons identified as authors of the code. All rights reserved.

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

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


revision 2022-03-06 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network.";
}

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

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

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

identity ineligible-originator {
  base ineligible-route-reason;
  description
    "Route was ineligible due to ORIGINATOR_ID. For example, update has local router as originator";
}
identity ineligible-confed {
  base ineligible-route-reason;
  description
    "Route was ineligible due to a loop in the AS_CONFED_SEQUENCE
    or AS_CONFED_SET attributes";
}

identity bgp-not-selected-bestpath {
  description
    "Base identity for indicating reason a route was was not
    selected by BGP route selection algorithm";
  reference
    "RFC 4271 - 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 IBGP, 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.";
  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 applying import policies.";
}

}
<CODE BEGINS> file "ietf-bgp-rib-attributes@2022-03-06.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";
  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 juniper.net).";

  description "This submodule contains common data definitions for BGP
               attributes for use in BGP RIB tables."

  Copyright (c) 2021 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject to
  the license terms contained in, the Simplified BSD License set
  forth in Section 4.c of the IETF Trust’s Legal Provisions
  Relating to IETF Documents

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


revision 2022-03-06 {
  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-top {
  description
    "Unknown path attributes that are not expected to be shared
across route entries, common to LOC-RIB and Adj-RIB";
container unknown-attributes {
  description
  "Unknown path attributes that were received in the UPDATE message which contained the prefix.";

list unknown-attribute {
  key "attr-type";
  description
  "This list contains received attributes that are unrecognized or unsupported by the local router. The list may be empty.";

  leaf optional {
    type boolean;
    description
    "Defines whether the attribute is optional (if set to true) or well-known (if set to false). Set in the high-order bit of the BGP attribute flags octet.";
    reference
    "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
  }

  leaf transitive {
    type boolean;
    description
    "Defines whether an optional attribute is transitive (if set to true) or non-transitive (if set to false). For well-known attributes, the transitive flag must be set to true. Set in the second high-order bit of the BGP attribute flags octet.";
    reference
    "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
  }

  leaf partial {
    type boolean;
    description
    "Defines whether the information contained in the optional transitive attribute is partial (if set to true) or complete (if set to false). For well-known attributes and for optional non-transitive attributes, the partial flag must be set to false. Set in the third high-order bit of the BGP attribute flags octet.";
    reference
    "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
  }
}
leaf extended {
  type boolean;
  description
  "Defines whether the attribute length is one octet
  (if set to false) or two octets (if set to true). Set in
  the fourth high-order bit of the BGP attribute flags
  octet.";
  reference
  "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
}

leaf attr-type {
  type uint8;
  description
  "1-octet value encoding the attribute type code";
  reference
  "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
}

leaf attr-len {
  type uint16;
  description
  "One or two octet attribute length field indicating the
  length of the attribute data in octets. If the Extended
  Length attribute flag is set, the length field is 2
  octets, otherwise it is 1 octet";
  reference
  "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
}

leaf attr-value {
  type binary {
    length "0..65535";
  }
  description
  "Raw attribute value, not including the attribute
  flags, type, or length. The maximum length
  of the attribute value data is 2^16-1 per the max value
  of the attr-len field (2 octets).";
  reference
  "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
}

}

grouping bgp-adj-rib-attr-state {
  description

"Path attributes that are not expected to be shared across route entries, specific to Adj-RIB";
leaf path-id {
  type uint32;
  description
    "When the BGP speaker supports advertisement of multiple paths for a prefix, the path identifier is used to uniquely identify a route based on the combination of the prefix and path id. In the Adj-RIB-In, the path-id value is the value received in the update message. In the Loc-RIB, if used, it should represent a locally generated path-id value for the corresponding route. In Adj-RIB-Out, it should be the value sent to a neighbor when add-paths is used, i.e., the capability has been negotiated."
  reference
    "RFC 7911: Advertisement of Multiple Paths in BGP";
}

<CODE BEGINS> file "ietf-bgp-rib-tables@2022-03-06.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.";
  }
  import ietf-bgp-types {
    prefix bt;
    reference

"RFC XXXX: BGP YANG Model for Service Provider Network."
}
include ietf-bgp-rib-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,
   Jeffrey Haas (jhaas at juniper.net).";

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

Copyright (c) 2021 IETF Trust and the persons identified as
authors of the code. All rights reserved.

 Redistribution and use in source and binary forms, with or
 without modification, is permitted pursuant to, and subject to
 the license terms contained in, the Simplified BSD License set
 forth in Section 4.c of the IETF Trust’s Legal Provisions
 Relating to IETF Documents

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

 ‘MAY’, and ‘OPTIONAL’ in this document are to be interpreted as
 described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
 they appear in all capitals, as shown here.";

revision 2022-03-06 {
   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 yang:timeticks;
  description
    "Timestamp when this path was last modified. The value is the timestamp in seconds relative to the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
}
leaf eligible-route {
  type boolean;
  description
    "Indicates that the route is eligible for selection for the best route in the Loc-Rib in BGP’s Decision Process.";
  reference
    "RFC 4271, Section 9.1.";
}
leaf ineligible-reason {
  type identityref {
    base ineligible-route-reason;
  }
  description
    "If the route is ineligible for selection for the best route in the Loc-Rib in BGP’s Decision process, this indicates the reason.";
  reference
    "RFC 4271, Section 9.1.";
}
}

grouping bgp-adj-rib-in-post-route-annotations-state {
  description
    "Data definitions for information attached to routes in the Adj-RIB-in post-policy table";
  leaf best-path {
    type boolean;
    description
      "Current path was selected as the best path.";
  }
}

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";
}
}

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/
        + "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";
    }
    description "Reference to the extended community attribute for the route.";
}

grouping bgp-loc-rib-common-keys {
    description "Common references used in keys for IPv4 and IPv6 LOC-RIB entries.";
    leaf origin {
        type union {
            type inet:ip-address;
            type identityref {
                base rt:routing-protocol;
            }
        }
    }
    description "Indicates the origin of the route. If the route is learned from a neighbor, this value is the neighbor address. If the route was injected or redistributed from another protocol, the origin indicates the source protocol for the route.";
    leaf path-id {
        type uint32;
    }
}
description
"If the route is learned from a neighbor, the path-id corresponds to the path-id for the route in the corresponding adj-rib-in-post table. If the route is injected from another protocol, or the neighbor does not support BGP add-paths, the path-id should be set to zero, also the default value.

However, YANG does not allow default values to be set for parameters that form the key, so a default value cannot be set here.";
}
}
grouping clear-routes {
  description
    "Action to clear BGP routes.";
  container clear-routes {
    if-feature "bt:clear-routes";
    action clear {
      input {
        leaf clear-at {
          type yang:date-and-time;
          description
            "The time, in the future when the clear operation will be initiated.";
        }
      }
      output {
        leaf clear-finished-at {
          type yang:date-and-time;
          description
            "The time when the clear operation finished.";
        }
      }
    }
  }
}
grouping ipv4-adj-rib-common {
  description
    "Common structural grouping for each IPv4 adj-RIB table.";
  container routes {
    config false;
    description
      "Enclosing container for list of routes in the routing
list route {
  key "prefix path-id";
  description "List of routes in the table, keyed by a combination of the route prefix and path-id to distinguish multiple routes received from a neighbor for the same prefix, e.g., when BGP add-paths is enabled."
  leaf prefix {
    type inet:ipv4-prefix;
    description "Prefix for the route."
  }
  uses bgp-adj-rib-attr-state;
  uses bgp-adj-rib-common-attr-refs;
  uses bgp-common-route-annotations-state;
  uses bgp-unknown-attr-top;
  uses rib-ext-route-annotations;
}

grouping ipv4-adj-rib-in-post {
  description "Common structural grouping for the IPv4 adj-rib-in post-policy table."
  container routes {
    config false;
    description "Enclosing container for list of routes in the routing table."
    list route {
      key "prefix path-id";
      description "List of routes in the table, keyed by a combination of the route prefix and path-id to distinguish multiple routes received from a neighbor for the same prefix, e.g., when BGP add-paths is enabled."
      leaf prefix {
        type inet:ipv4-prefix;
        description "Prefix for the route."
      }
      uses bgp-adj-rib-attr-state;
      uses bgp-adj-rib-common-attr-refs;
      uses bgp-common-route-annotations-state;
      uses bgp-adj-rib-in-post-route-annotations-state;
      uses bgp-unknown-attr-top;
    }
  }
}
uses rib-ext-route-annotations;
}
}
}

grouping ipv6-adj-rib-common {

description
"Common structural grouping for each IPv6 adj-RIB table."
container routes {

cfg false;

description
"Enclosing container for list of routes in the routing table."
list route {

key "prefix path-id";

description
"List of routes in the table."
leaf prefix {

type inet:ipv6-prefix;

description
"Prefix for the route."
}
uses bgp-adj-rib-attr-state;
uses bgp-adj-rib-common-attr-refs;
uses bgp-common-route-annotations-state;
uses bgp-unknown-attr-top;
uses rib-ext-route-annotations;
}
}

grouping ipv6-adj-rib-in-post {

description
"Common structural grouping for the IPv6 adj-rib-in post-policy table."
container routes {

cfg 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."
}
8. Contributors

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

9. Acknowledgements

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Pavan Beeram, Chris Chase, Ed Crabbe, Luyuan Fang, Bill Fenner, Akshay Gattani, Josh George, Vijay Gill, Matt John, Jeff Haas, Dhanendra Jain, Acee Lindem, Ina Minei, Carl Moberg, Ashok Narayanan, Einar Nilsen-Nygård, 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.

10. References

10.1. Normative references


10.2. Informative references


Appendix A. Examples

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

A.1. Creating BGP Instance

This example shows how to enable BGP for a IPv4 unicast address family.

[Note: \ line wrapping for formatting only]

```xml
<?xml version="1.0" encoding="UTF-8"?>
<routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
    <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>
                            <name
                                xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">bt:ip\ v4-unicast</name>
                        </afi-safi>
                    </afi-safis>
                </global>
            </bgp>
        </control-plane-protocol>
    </control-plane-protocols>
</routing>
```
A.2. Neighbor Address Family Configuration

This example shows how to configure a BGP neighbor, where the remote address is 192.0.2.1, the remote AS number is 64497, and the address family of the neighbor is IPv4 unicast. The neighbor is configured for route flap prevention and it set up for standard and large communities. In addition, BFD is configured at a neighbor level with a local multiplier of 2, a desired minimum transmit interval, and a required minimum receive interval of 3.3 ms.

[note: '\' line wrapping for formatting only]

<!--
This example shows a neighbor configuration with damping.
-->

<?xml version="1.0" encoding="UTF-8"?>
<routing
 xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
xm
<control-plane-protocols>
<control-plane-protocol>
<
type
type>
<name>BGP</name>
<bgp
 xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
<global>
<as>64496</as>
<afi-safis>
<afi-safi>
<name>bt:ipv4-unicast</name>
</afi-safi>
</afi-safis>
</global>
<neighbors>
<neighbor>
<remote-address>192.0.2.1</remote-address>
<peer-as>64497</peer-as>
<route-flap-damping>
<enable>true</enable>
<suppress-above>4.0</suppress-above>
<reuse-above>3.0</reuse-above>
<max-flap>15.0</max-flap>
<reach-decay>100</reach-decay>
<unreach-decay>500</unreach-decay>
<keep-history>1000</keep-history>
</route-flap-damping>
A.3. IPv6 Neighbor Configuration

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

[<xml version="1.0" encoding="UTF-8">
<key-chains
  xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain"
<key-chain
  <name>bgp-key-chain</name>
</key-chain>
</key-chains>
<routing
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
<control-plane-protocols>
  <control-plane-protocol
    <type
    <name>name:BGP</name>
  <bgp>
A.4. VRF Configuration

This example shows how BGP can be configured for two VRFs, red and blue. In this case, the two network instances share a common AS, and distinguish between the instances using the router id.
<?xml version="1.0" encoding="UTF-8"?>
<network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
        <name>vrf-red</name>
        <vrf-root>
            <routing
                xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
                <router-id>192.0.2.1</router-id>
                <control-plane-protocols>
                    <control-plane-protocol>
                        <type
                        <name>BGP</name>
                        <bgp
                            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
                            <global>
                                <as>64496</as>
                                <afi-safis>
                                    <afi-safi>
                                        <name
                                            xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">bt:ipv4-unicast</name>
                                    </afi-safi>
                                </afi-safis>
                            </global>
                        </bgp>
                    </control-plane-protocol>
                </control-plane-protocols>
            </routing>
        </vrf-root>
    </network-instance>
    <network-instance>
        <name>vrf-blue</name>
        <vrf-root>
            <routing
                xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
                <router-id>192.0.2.2</router-id>
                <control-plane-protocols>
                    <control-plane-protocol>
                        <type
                        <name>BGP</name>
                        <bgp
                            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
                            <global>
                                <as>64496</as>
                                <afi-safis>
                                    <afi-safi>
                                        <name
                                            xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">bt:ipv4-unicast</name>
                                    </afi-safi>
                                </afi-safis>
                            </global>
                        </bgp>
                    </control-plane-protocol>
                </control-plane-protocols>
            </routing>
        </vrf-root>
    </network-instance>
</network-instances>
<name>BGP</name>
<bgp xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
  <global>
    <as>64496</as>
    <afi-safis>
      <afi-safi>
        <name xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">
          bt:ipv4-unicast
        </name>
      </afi-safi>
    </afi-safis>
  </global>
</bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</vrf-root>
</network-instance>
</network-instances>

A.5. BGP Policy

Routing policy using community value involves configuring rules to match community values in the inbound or outbound direction. In this example, which is heavily borrowed from the example on the Cisco community page, we look at "match community exact" match, which happens only when BGP updates have the same community values as specified in the community list.

The topology in this example consists of three routers, R1, R2, and R3, configured with AS value of 1, 2 and 3 respectively. R1 advertises 5 prefixes to R2 and R3, as shown below.

* 1.1.1.1/32 and 2.2.2.2/32 with community 11:11
* 3.3.3.3/32 and 4.4.4.4/32 with community 11:11 and 22:22
* 5.5.5.5/32 with community 33.33
Route Policy TO_R2 defines the policy that R1 uses in route updates towards R2. It consists of three statements, statement 10 that has an exact match rule for the prefix list L0andL1, and a set-community action of add for 11:11. The second statement, statement 20, consists of an exact match rule for prefix list L2andL3, with a set community action of remove for 11:11 22:22. The final statement, statement 30, consists of an exact match rule for prefix list L4, with a set community action of replace for 33:33.

[note: \ line wrapping for formatting only]

```xml
<xml version="1.0" encoding="UTF-8"?
<routing-policy
 xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">
<defined-sets>
<prefix-sets>
<prefix-set>
<name>L0andL1</name>
<mode>ipv4</mode>
<prefixes>
<prefix-list>
<ip-prefix>1.1.1.1/32</ip-prefix>
<mask-length-lower>32</mask-length-lower>
<mask-length-upper>32</mask-length-upper>
</prefix-list>
<prefix-list>
<ip-prefix>2.2.2.2/32</ip-prefix>
<mask-length-lower>32</mask-length-lower>
<mask-length-upper>32</mask-length-upper>
</prefix-list>
</prefixes>
</prefix-set>
<prefix-set>
<name>L2andL3</name>
<mode>ipv4</mode>
<prefixes>
<prefix-list>
<ip-prefix>3.3.3.3/32</ip-prefix>
<mask-length-lower>32</mask-length-lower>
<mask-length-upper>32</mask-length-upper>
</prefix-list>
<prefix-list>
<ip-prefix>4.4.4.4/32</ip-prefix>
<mask-length-lower>32</mask-length-lower>
<mask-length-upper>32</mask-length-upper>
</prefix-list>
</prefixes>
</prefix-set>
</prefix-sets>
</routing-policy>
```
<prefix-set>
  <name>L4</name>
  <mode>ipv4</mode>
  <prefixes>
    <prefix-list>
      <ip-prefix>5.5.5.5/32</ip-prefix>
      <mask-length-lower>32</mask-length-lower>
      <mask-length-upper>32</mask-length-upper>
    </prefix-list>
  </prefixes>
</prefix-set>
</prefix-sets>
</defined-sets>
<policy-definitions>
  <policy-definition>
    <name>TO_R2</name>
    <statements>
      <statement>
        <name>10</name>
        <conditions>
          <match-prefix-set>
            <prefix-set>L0andL1</prefix-set>
          </match-prefix-set>
        </conditions>
        <actions>
          <bgp-actions xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">
            <set-community>
              <options>add</options>
              <communities>11:11</communities>
            </set-community>
          </bgp-actions>
        </actions>
      </statement>
      <statement>
        <name>20</name>
        <conditions>
          <match-prefix-set>
            <prefix-set>L2andL3</prefix-set>
          </match-prefix-set>
        </conditions>
        <actions>
          <bgp-actions xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">
            <set-community>
              <options>remove</options>
            </set-community>
          </bgp-actions>
        </actions>
      </statement>
    </statements>
  </policy-definition>
</policy-definitions>
Appendix B. How to add a new AFI and Augment a Module

This section explains how a new AFI can be defined in a new module and how that module can then be augmented. Assume that the new AFI being defined is called 'foo' which extends the base identity of 'afi-safi-type', and the augmentation is to add a new container for 'foo' under two different XPaths. The example shows how the base identity can be extended to add this new AFI, and then use the augmented containers be used to add 'foo' specific information.

module example-newafi-bgp {
  yang-version 1.1;
  namespace "http://example.com/ns/example-newafi-bgp";
  prefix example-newafi-bgp;

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

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

  ...
import ietf-bgp {
  prefix "bgp";
  reference
    "RFC XXXX: 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 2022-03-06 {
  description
    "Creating new AFI and using in this model";

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

identity foo {
  base bt:afi-safi-type;
  description
    "New AFI type foo.";
}

augment "/rt:routing/rt:control-plane-protocols/" +
  "rt:control-plane-protocol/bgp:bgp:bgp:global/" +
  "bgp:afi-safis/bgp:afi-safi" {
  when "derived-from-or-self(bgp:name, 'foo')" {
    description
      "This augmentation is valid for a AFI/SAFI instance
      of 'foo';";
  }
  container foo {
    description
      "Container to add 'foo' specific AFI/SAFI information.
      First add the common stuff.";
    uses bgp:mp-all-afi-safi-common;
  }
}
augment "/rt:routing/rt:control-plane-protocols/" + 
  "rt:control-plane-protocol/bgp:bgp/" + 
  "bgp:rib/bgp:afi-safis/bgp:afi-safi" {
when "derived-from-or-self(bgp:name, 'foo')" {
  description
  "This augmentation is valid for a AFI/SAFI instance of 'foo';"
}
}

container foo {
  description
  "Container to add 'foo' rib specific information. First add the common stuff."
  container loc-rib {
    config false;
    description
    "Container for the 'foo' BGP LOC-RIB data."
    container routes {
      description
      "Enclosing container for list of routes in the routing table."
      list route {
        key "prefix origin path-id";
        description
        "List of routes in the table, keyed by the route prefix, the route origin, and path-id. The route origin can be either the neighbor address from which the route was learned, or the source protocol that injected the route. The path-id distinguishes routes for the same prefix received from a neighbor (e.g., if add-paths is enabled)."
        leaf prefix {
          type inet:ip-address;
          description
          "The 'foo' prefix corresponding to the route."
        }
        uses bgp:bgp-loc-rib-common-keys;
        uses bgp:bgp-loc-rib-common-attr.refs;
        uses bgp:bgp-common-route-annotations-state;
        uses bgp:bgp-unknown-attr-top;
        uses bgp:rib-ext-route-annotations;
      }
      uses bgp:clear-routes;
    }
  }
}

container neighbors {
    config false;
    description
        "Enclosing container for neighbor list.";
    list neighbor {
        key "neighbor-address";
        description
            "List of neighbors (peers) of the local BGP speaker.";
        leaf neighbor-address {
            type inet:ip-address;
            description
                "IP address of the BGP neighbor or peer.";
        }
    }
    container adj-rib-in-pre {
        description
            "Per-neighbor table containing the NLRI updates
            received from the neighbor before any local input
            policy rules or filters have been applied. This can
            be considered the ‘raw’ updates from the neighbor.";
        uses bgp:ipv4-adj-rib-common;
    }
    container adj-rib-in-post {
        description
            "Per-neighbor table containing the paths received from
            the neighbor that are eligible for best-path selection
            after local input policy rules have been applied.";
        uses bgp:ipv4-adj-rib-in-post;
    }
    container adj-rib-out-pre {
        description
            "Per-neighbor table containing paths eligible for
            sending (advertising) to the neighbor before output
            policy rules have been applied.";
        uses bgp:ipv4-adj-rib-common;
    }
    container adj-rib-out-post {
        description
            "Per-neighbor table containing paths eligible for
            sending (advertising) to the neighbor after output
            policy rules have been applied.";
        uses bgp:ipv4-adj-rib-common;
    }
}
Appendix C. How to deviate a module

This example shows how the BGP can be deviated to indicate two nodes that the particular implementation is choosing not to support.

module example-newco-bgp {
  yang-version 1.1;
  namespace "http://example.com/ns/example-newco-bgp";
  prefix example-newco-bgp;

  import ietf-bgp {
    prefix "bgp";
  }

  organization
    "Newco model group.";

  contact
    "abc@newco.com";
  description
    "This YANG module deviates IETF BGP YANG module.";

  revision 2022-03-06 {
    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;
  }
}

Appendix D. Complete configuration tree diagram

Here is a complete tree diagram for the configuration and operational part of the model.
module: ietf-bgp

augment /rt:routing/rt:control-plane-protocols
   /rt:control-plane-protocol:
   
      +++--rw bgp
      
         +++--rw global!
         
            +++--rw as inet:as-number
            +--rw identifier? yang:dotted-quad
            +--rw distance
               +--rw external? uint8
               +--rw internal? uint8
            
            +++--rw confederation
               +--rw enabled? boolean
               +--rw identifier? inet:as-number
               +--rw member-as* inet:as-number
            
            +++--rw graceful-restart (bt:graceful-restart)?
               +--rw enabled? boolean
               +--rw restart-time? uint16
               +--rw stale-routes-time? uint32
               +--rw helper-only? boolean
            
            +++--rw use-multiple-paths
               +--rw enabled? boolean
               +--rw ebgp
               
                  +++--rw allow-multiple-as? boolean
                  +--rw maximum-paths? uint32
               
               +--rw ibgp
               
                  +++--rw maximum-paths? uint32
               
            +++--rw route-selection-options
               
                  +++--rw always-compare-med? boolean
                  +--rw ignore-as-path-length? boolean
                  +--rw external-compare-router-id? boolean
                  +--rw advertise-inactive-routes? boolean
                  +--rw enable-aigp? boolean
                  +--rw ignore-next-hop-igp-metric? boolean
                  +--rw enable-med? boolean
               
                  +++--rw med-plus-igp
                  
                     +--rw enabled? boolean
                     +--rw igp-multiplier? uint16
                     +--rw med-multiplier? uint16
               
               +++--rw afi-safis
               
                  +++--rw afi-safi* [name]
               
                     +++--rw name identityref
                     +--rw enabled? boolean
                     +--ro total-paths? uint32
                     +--ro total-prefixes? uint32
                     
                     +++--rw graceful-restart (bt:graceful-restart)?
                     | +--rw enabled? boolean
                     
                     +++--rw route-selection-options
++-rw always-compare-med?  boolean
++-rw ignore-as-path-length?  boolean
++-rw external-compare-router-id?  boolean
++-rw advertise-inactive-routes?  boolean
++-rw enable-aigp?  boolean
++-rw ignore-next-hop-igp-metric?  boolean
++-rw enable-med?  boolean
++-rw med-plus-igp
    +++-rw enabled?  boolean
    +++-rw igp-multiplier?  uint16
    +++-rw med-multiplier?  uint16
+++-rw use-multiple-paths
    ++-rw enabled?  boolean
    ++-rw ebgp
        +++-rw allow-multiple-as?  boolean
        +++-rw maximum-paths?  uint32
    ++-rw ibgp
        +++-rw maximum-paths?  uint32
+++-rw apply-policy
    ++-rw import-policy*  leafref
    ++-rw default-import-policy?  default-policy-type
    ++-rw export-policy*  leafref
    ++-rw default-export-policy?  default-policy-type
+++-rw ipv4-unicast
    ++-rw prefix-limit
        +++-rw max-prefixes?  uint32
        +++-rw shutdown-threshold-pct?  rt-types:percentage
        +++-rw restart-timer?  uint32
    ++-rw send-default-route?  boolean
+++-rw ipv6-unicast
    ++-rw prefix-limit
        +++-rw max-prefixes?  uint32
        +++-rw shutdown-threshold-pct?  rt-types:percentage
        +++-rw restart-timer?  uint32
    ++-rw send-default-route?  boolean
+++-rw ipv4-labeled-unicast
    ++-rw prefix-limit
        +++-rw max-prefixes?  uint32
        +++-rw shutdown-threshold-pct?  rt-types:percentage
        +++-rw restart-timer?  uint32
+++-rw ipv6-labeled-unicast
    ++-rw prefix-limit
        +++-rw max-prefixes?  uint32
        +++-rw shutdown-threshold-pct?  rt-types:percentage
| +--rw restart-timer?            uint32
| ++-rw 13vpn-ipv4-unicast
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| ++-rw 13vpn-ipv6-unicast
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| ++-rw 13vpn-ipv4-multicast
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| ++-rw 13vpn-ipv6-multicast
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| ++-rw l2vpn-vpls
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| ++-rw l2vpn-evpn
|    ++-rw prefix-limit
|    |    ++-rw max-prefixes?             uint32
|    |    ++-rw shutdown-threshold-pct?   rt-types:percentage
|    |    ++-rw restart-timer?            uint32
| +--rw apply-policy
|    ++-rw import-policy*           leafref
|    ++-rw default-import-policy?   default-policy-type
|    ++-rw export-policy*           leafref
|    ++-rw default-export-policy?   default-policy-type
|    ++-ro total-paths?             uint32
|    ++-ro total-prefixes?          uint32

++-rw neighbors
|    ++-rw neighbor* [remote-address]
|    |    ++-rw remote-address               inet:ip-address
|    |    ++-ro local-address?               inet:ip-address
++-ro local-port?                    inet:port-number
++-ro remote-port?                  inet:port-number
++-ro peer-type?                    bt:peer-type
++-rw peer-group?
    -> ../../../peer-groups/peer-group/name
++-ro identifier?                   yang:dotted-quad
++-rw enabled?                       boolean
++-rw secure-session-enable?         boolean
++-rw secure-session
     ++-rw (option)?
        ++-:(ao)
         ++-rw enable-ao?                boolean
         ++-rw send-id?                   uint8
         ++-rw recv-id?                   uint8
         ++-rw include-tcp-options?       boolean
         ++-rw accept-ao-mismatch?        boolean
         ++-rw ao-keychain?
             key-chain:key-chain-ref
        ++-:(md5)
         ++-rw enable-md5?                 boolean
         ++-rw md5-keychain?
             key-chain:key-chain-ref
     ++-rw ttl-security?                uint8
        (bt:ttl-security)?
++-rw peer-as?                      inet:as-number
++-rw local-as?                      inet:as-number
++-rw remove-private-as?
    bt:remove-private-as-option
++-rw route-flap-damping (bt:damping)?
    ++-rw enable?                      boolean
    ++-rw suppress-above?              decimal64
    ++-rw reuse-above?                 decimal64
    ++-rw max-flap?                    decimal64
    ++-rw reach-decay?                 uint32
    ++-rw unreachable-decay?           uint32
    ++-rw keep-history?                uint32
++-rw send-community*               identityref
    (bt:send-communities)?
++-rw description?                   string
++-rw timers
    ++-rw connect-retry-interval?      uint16
    ++-rw hold-time?                   uint16
    ++-rw keepalive?                   uint16
    ++-rw min-as-origination-interval? uint16
    ++-rw min-route-advertisement-interval? uint16
++-rw transport
    ++-rw tcp-mss?                     uint16
    ++-rw mtu-discovery?               boolean
`---rw passive-mode? boolean
---rw local-address? union
  +--rw md5-auth-password? string
  +--rw bfd (bt:bfd)?
      +--rw enabled? boolean
      +--rw local-multiplier? multiplier
  +--rw (interval-config-type)?
      +--:(tx-rx-intervals)
          +--rw desired-min-tx-interval? uint32
          +--rw required-min-rx-interval? uint32
      +--:(single-interval)
          +--rw (single-minimum-interval)?
              +--rw min-interval? uint32
---rw graceful-restart (bt:graceful-restart)?
  +--rw enabled? boolean
  +--rw restart-time? uint16
  +--rw stale-routes-time? uint32
  +--rw helper-only? boolean
  +--rw peer-restart-time? uint16
  +--ro peer-restarting? boolean
  +--ro local-restarting? boolean
  +--ro mode? enumeration
---rw logging-options
  +--rw log-neighbor-state-changes? boolean
---rw ebgp-multihop
  +--rw enabled? boolean
  +--rw multihop-ttl? uint8
---rw route-reflector
  +--rw cluster-id? bt:rr-cluster-id-type
  +--rw no-client-reflect? boolean
  +--rw client? boolean
---rw as-path-options
  +--rw allow-owner-as? uint8
  +--rw replace-peer-as? boolean
---rw add-paths (bt:add-paths)?
  +--rw receive? boolean
  +--rw (send)?
      +--:(max)
          +--rw max? uint8
      +--:(all)
          +--rw all? empty
  +--rw eligible-prefix-policy? leafref
---rw use-multiple-paths
  +--rw enabled? boolean
  +--rw ebgp
      +--rw allow-multiple-as? boolean
---rw apply-policy
  +--rw import-policy? leafref
Internet-Draft               BGP YANG Model                   March 2022

---rw max-prefixes?          uint32
---rw shutdown-threshold-pct?  rt-types:percentage
   ---rw restart-timer?    uint32

---rw max-prefixes?     uint32
---rw shutdown-threshold-pct?  rt-types:percentage
   ---rw restart-timer?    uint32

---rw use-multiple-paths
   ---rw enabled?   boolean
```yang
define module bgp {
  include iana-asn;

  namespace "http://www.example.com/bgp"
  prefix bgp

  module ebgp {
    import bgp
    import iana-asn
    import iana-protocol-number

    leaf allow-multiple-as? {
      default "false"
    }

    leaf session-state? {
      type enum
      enum "established"
      enum "initial"
      enum "outdated"
      enum "closed"
    }

    leaf last-established? {
      type date-and-time
    }

    leaf negotiated-capabilities* {
      type identityref
      mandatory
    }

    leaf negotiated-hold-time? {
      type uint16
    }

    leaf last-error? {
      type binary
    }

    leaf fsm-established-time? {
      type gauge32
    }

    leaf treat-as-withdraw? {
      default "false"
    }

    leaf erroneous-update-messages? {
      type uint32
    }

    leaf bfd {
      leaf bfd-enabled? {
        default "false"
      }
      leaf bfd-local-multiplier? {
        type multiplier
      }
      leaf bfd-interval-config-type? {
        type enum
        enum "single-interval"
      }
      leaf bfd-desired-min-tx-interval? {
        type uint32
      }
      leaf bfd-required-min-rx-interval? {
        type uint32
      }
      leaf bfd-single-interval-minimum-interval? {
        leaf bfd-min-interval? {
          type uint32
        }
      }
      leaf bfd-peer-fsm-established-transitions? {
        type counter64
      }
      leaf bfd-local-fsm-established-transitions? {
        type counter32
      }
      leaf bfd-messages? {
        leaf bfd-in-total-messages? {
          type counter32
        }
        leaf bfd-out-total-messages? {
          type counter32
        }
        leaf bfd-in-update-elapsed-time? {
          type gauge32
        }
        leaf bfd-sent {
          leaf bfd-updates-received? {
            type uint64
          }
          leaf bfd-updates-sent? {
            type uint64
          }
          leaf bfd-messages-received? {
            type uint64
          }
          leaf bfd-messages-sent? {
            type uint64
          }
          leaf bfd-notification? {
            type uint64
          }
        }
        leaf bfd-received {
          leaf bfd-updates-received? {
            type uint64
          }
          leaf bfd-updates-sent? {
            type uint64
          }
          leaf bfd-messages-received? {
            type uint64
          }
          leaf bfd-messages-sent? {
            type uint64
          }
          leaf bfd-notification? {
            type uint64
          }
        }
      }
      leaf bfd-queues {
        leaf bfd-input? {
          type uint32
        }
        leaf bfd-output? {
          type uint32
        }
      }
      leaf bfd-clear {
        leaf bfd-clear-statistics? {
          leaf bfd-clear-at? {
            type date-and-time
          }
        }
      }
    }
  }
}
```
++-n established
  |  +-- remote-address?  -> ../../neighbor/remote-address
  |  +-- last-error?     -> ../../neighbor/last-error
  |  +-- session-state?  -> ../../neighbor/session-state
++-n backward-transition
  |  +-- remote-addr?    -> ../../neighbor/remote-address
  |  +-- last-error?     -> ../../neighbor/last-error
  |  +-- session-state?  -> ../../neighbor/session-state
+++x clear (bt:clear-neighbors)?
  +++-w input
    |  +++-w (operation)?
    |     |  +++-w (operation-admin)
    |     |     |  +++-w admin?        empty
    |     |  +++-w (operation-hard)
    |     |     |  +++-w hard?         empty
    |     |  +++-w (operation-soft)
    |     |     |  +++-w soft?         empty
    |     |  +++-w (operation-soft-inbound)
    |     |     |  +++-w soft-inbound? empty {bt:route-refresh}?  
    |     |  +++-w clear-at?     yang:date-and-time
  +++-ro output
    |  +++-ro clear-finished-at?  yang:date-and-time
+++rw peer-groups
  +++rw peer-group* [name]
    |  +++rw name            string
    |  +++rw secure-session-enable? boolean
    |  +++rw secure-session
      |  +++-rw (option)?
      |     |  +++-rw enable-ao?    boolean
      |     |  +++-rw send-id?      uint8
      |     |  +++-rw recv-id?      uint8
      |     |  +++-rw include-tcp-options? boolean
      |     |  +++-rw accept-ao-mismatch? boolean
      |     |  +++-rw ao-keychain?
      |     |     |  key-chain:key-chain-ref
      |     |  +++-rw enable-md5?    boolean
      |     |  +++-rw md5-keychain?
      |     |     |  key-chain:key-chain-ref
      |     |  +++-rw sa?           string
      |     |  +++-rw ttl-security? uint8 {bt:ttl-security}?
      |     |  +++-rw peer-as?      inet:as-number
      |     |  +++-rw local-as?     inet:as-number
      |     |  +++-rw remove-private-as?
      |     |     |  bt:remove-private-as-option
      |     |  +++-rw route-flap-damping (bt:damping)?
Internet-Draft               BGP YANG Model                   March 2022

---rw no-client-reflect? boolean
---rw client? boolean
---rw as-path-options
  ---rw allow-own-as? uint8
  ---rw replace-peer-as? boolean
---rw add-paths {bt:add-paths}?
  ---rw receive? boolean
  ---rw (send)?
    +-:(max)
      ---rw max? uint8
    +-:(all)
      ---rw all? empty
  ---rw eligible-prefix-policy? leafref
---rw use-multiple-paths
  ---rw enabled? boolean
  ---rw ebgp
    ---rw allow-multiple-as? boolean
---rw apply-policy
  ---rw import-policy* leafref
  ---rw default-import-policy? default-policy-type
  ---rw export-policy* leafref
  ---rw default-export-policy? default-policy-type
---rw afi-safis
  ---rw afi-safi* [name]
    ---rw name identityref
    ---rw enabled? boolean
    ---rw graceful-restart {bt:graceful-restart}?
      ---rw enabled? boolean
    ---rw use-multiple-paths
      ---rw enabled? boolean
      ---rw ebgp
        ---rw allow-multiple-as? boolean
    ---rw apply-policy
      ---rw import-policy* leafref
      ---rw default-import-policy? default-policy-type
      ---rw export-policy* leafref
      ---rw default-export-policy? default-policy-type
    ---rw ipv4-unicast
      ---rw prefix-limit
        ---rw max-prefixes? uint32
        ---rw shutdown-threshold-pct?
          rt-types:percentage
        ---rw restart-timer? uint32
      ---rw send-default-route? boolean
    ---rw ipv6-unicast
      ---rw prefix-limit
---rw max-prefixes?     uint32
---rw shutdown-threshold-pct?
  |   rt-types:percentage
---rw restart-timer?    uint32
---rw send-default-route?  boolean

---rw ipv4-labeled-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw ipv6-labeled-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw 13vpn-ipv4-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw 13vpn-ipv6-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw 13vpn-ipv4-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw 13vpn-ipv6-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l2vpn-vpls
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l2vpn-evpn
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-labeled-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-ipv4-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-ipv6-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-labeled-unicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-ipv4-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l3vpn-ipv6-multicast
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l2vpn-vpls
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32

---rw l2vpn-evpn
  +--rw prefix-limit
  |   ---rw max-prefixes?     uint32
  |   ---rw shutdown-threshold-pct?
  |     |       rt-types:percentage
  |   ---rw restart-timer?    uint32
Internet-Draft               BGP YANG Model                   March 2022
[81x692]Internet-Draft               BGP YANG Model                   March 2022

---ro large-community* [index]
   +--ro index          uint64
   +--ro large-community* bt:large-community-type
---ro afi-safis
   +--ro afi-safi* [name]
      +--ro name identityref
---ro ipv4-unicast
   +--ro loc-rib
      +--ro routes
         +--ro route* [prefix origin path-id]
            +--ro prefix
               |    inet:ipv4-prefix
            +--ro origin union
            +--ro path-id uint32
            +--ro attr-index? leafref
            +--ro community-index? leafref
            +--ro ext-community-index? leafref
            +--ro last-modified? yang:timeticks
            +--ro eligible-route? boolean
            +--ro ineligible-reason? identityref
         +--ro unknown-attributes
            +--ro unknown-attribute* [attr-type]
               +--ro optional? boolean
               +--ro transitive? boolean
               +--ro partial? boolean
               +--ro extended? boolean
               +--ro attr-type uint8
               +--ro attr-len? uint16
               +--ro attr-value? binary
               +--ro reject-reason? union
      +--ro neighbors
         +--ro neighbor* [neighbor-address]
            +--ro neighbor-address inet:ip-address
         +--ro adj-rib-in-pre
            +--ro routes
               +--ro route* [prefix path-id]
                  +--ro prefix
                     |    inet:ipv4-prefix
                  +--ro path-id uint32
                  +--ro attr-index? leafref
                  +--ro community-index? leafref
                  +--ro ext-community-index? leafref
                  +--ro last-modified? yang:timeticks
                  +--ro eligible-route? boolean
                  +--ro ineligible-reason? boolean
                  +--ro ineligable-reason?
identityref

---ro unknown-attributes
  ---ro unknown-attribute*
    [attr-type]
      ---ro optional? boolean
      ---ro transitive? boolean
      ---ro partial? boolean
      ---ro extended? boolean
      ---ro attr-type uint8
      ---ro attr-len? uint16
      ---ro attr-value? binary
      ---ro reject-reason? union

---ro clear-routes (bt:clear-routes)?
  +++-x clear
    +++-w input
      +++-w clear-at?
       .yang:date-and-time
    ---ro output
      +++-ro clear-finished-at?
       .yang:date-and-time

---ro adj-rib-in-post

---ro routes
  ---ro route* [prefix path-id]
    ---ro prefix
      inet:ipv4-prefix
    ---ro path-id uint32
    ---ro attr-index? leafref
    ---ro community-index? leafref
    ---ro ext-community-index? leafref
    ---ro last-modified?
      yang:timeticks
    ---ro eligible-route?
      boolean
    ---ro ineligible-reason?
      identityref
    ---ro best-path?
      boolean
    ---ro unknown-attributes
      ---ro unknown-attribute*
        [attr-type]
          ---ro optional? boolean
          ---ro transitive? boolean
          ---ro partial? boolean
          ---ro extended? boolean
          ---ro attr-type uint8
          ---ro attr-len? uint16
          ---ro attr-value? binary
          ---ro reject-reason? union
++-ro clear-routes {bt:clear-routes}?
  +++-x clear
    +++-w input
      +++-w clear-at?
        yang:date-and-time
    +++-ro output
      +++-ro clear-finished-at?
        yang:date-and-time
  +++-ro adj-rib-out-pre
    +++-ro routes
      +++-ro route* [prefix path-id]
        +++-ro prefix
          inet:ipv4-prefix
        +++-ro path-id      uint32
        +++-ro attr-index?  leafref
        +++-ro community-index?  leafref
        +++-ro ext-community-index?  leafref
        +++-ro last-modified?
          yang:timeticks
        +++-ro eligible-route?
          boolean
        +++-ro ineligible-reason?
          identityref
        +++-ro unknown-attributes
          +++-ro unknown-attribute*
            [attr-type]
              +++-ro optional?  boolean
              +++-ro transitive?  boolean
              +++-ro partial?  boolean
              +++-ro extended?  boolean
              +++-ro attr-type  uint8
              +++-ro attr-len?  uint16
              +++-ro attr-value?  binary
              +++-ro reject-reason?  union
        +++-ro clear-routes {bt:clear-routes}?
          +++-x clear
            +++-w input
              +++-w clear-at?
                yang:date-and-time
            +++-ro output
              +++-ro clear-finished-at?
                yang:date-and-time
  +++-ro adj-rib-out-post
    +++-ro routes
      +++-ro route* [prefix path-id]
        +++-ro prefix
          inet:ipv4-prefix
        +++-ro path-id      uint32
+++ro attr-index?  leafref
+++ro community-index?  leafref
+++ro ext-community-index?  leafref
+++ro last-modified?
    +---yang:timeticks
+++ro eligible-route?  boolean
+++ro ineligible-reason?  identityref
+++ro unknown-attributes
    +++ro unknown-attribute*[attr-type]
      +++ro optional?  boolean
      +++ro transitive?  boolean
      +++ro partial?  boolean
      +++ro extended?  boolean
      +++ro attr-type  uint8
      +++ro attr-len?  uint16
      +++ro attr-value?  binary
    ++---ro reject-reason?  union
+++ro clear-routes {bt:clear-routes}?  
    ++---x clear
      ++---w input
        ++---w clear-at?
          +---yang:date-and-time
        ++---ro output
          +---ro clear-finished-at?
            +---yang:date-and-time
+++ro ipv6-unicast
+++ro loc-rib
    +++ro routes
      +++ro route*[prefix origin path-id]
        +++ro prefix
          +---inet:ipv6-prefix
            +++ro origin  union
            +++ro path-id  uint32
            +++ro attr-index?  leafref
            +++ro community-index?  leafref
            +++ro ext-community-index?  leafref
            +++ro last-modified?
              +---yang:timeticks
            +++ro eligible-route?  boolean
            +++ro ineligible-reason?  identityref
              +++ro unknown-attributes
                +++ro unknown-attribute*[attr-type]
                  +++ro optional?  boolean
                  +++ro transitive?  boolean
                  +++ro partial?  boolean
Internet-Draft               BGP YANG Model                   March 2022

|        |     +--ro extended?     boolean
|        |     +--ro attr-type     uint8
|        |     +--ro attr-len?     uint16
|        |     +--ro attr-value?   binary
|        |     +--ro reject-reason?         union
|        +--ro neighbors
|        +--ro neighbor* [neighbor-address]
|        +--ro neighbor-address     inet:ip-address
|        +--ro adj-rib-in-pre
|        +--ro routes
|        |        +--ro route* [prefix path-id]
|        |        +--ro prefix
|        |        |        inet:ipv6-prefix
|        |        +--ro path-id       uint32
|        |        +--ro attr-index?   leafref
|        |        +--ro community-index?     leafref
|        |        +--ro ext-community-index?     leafref
|        |        +--ro last-modified?
|        |        |        yang:timeticks
|        |        +--ro eligible-route?
|        |        |        boolean
|        |        +--ro ineligible-reason?     identityref
|        +--ro unknown-attributes
|        |        +--ro unknown-attribute* [attr-type]
|        |        +--ro optional?     boolean
|        |        +--ro transitive?    boolean
|        |        +--ro partial?      boolean
|        |        +--ro extended?     boolean
|        |        +--ro attr-type     uint8
|        |        +--ro attr-len?     uint16
|        |        +--ro attr-value?   binary
|        |        +--ro reject-reason?         union
|        +--ro clear-routes {bt:clear-routes}?
|        |        +---x clear
|        |        |        +---w input
|        |        |        |        +---w clear-at?
|        |        |        |        |        yang:date-and-time
|        |        |        +--ro output
|        |        |        +--ro clear-finished-at?
|        |        |        |        yang:date-and-time
|        +--ro adj-rib-in-post
|        +--ro routes
|        |        +--ro route* [prefix path-id]
|        |        +--ro prefix
|        |        |        inet:ipv6-prefix
|        |        +--ro path-id       uint32

++-ro attr-index?   leafref
++-ro community-index? leafref
++-ro ext-community-index? leafref
++-ro last-modified?
|   yang:timeticks
++-ro eligible-route? boolean
++-ro ineligible-reason? identityref
++-ro best-path? boolean
++-ro unknown-attributes
|   ++-ro unknown-attribute*
|     [attr-type]
|     +--ro optional? boolean
|     +--ro transitive? boolean
|     +--ro partial? boolean
|     +--ro extended? boolean
|     +--ro attr-type uint8
|     +--ro attr-len? uint16
|     +--ro attr-value? binary
++-ro reject-reason? union
++-ro clear-routes (bt:clear-routes)?
|   +++-x clear
|     +++-w input
|     |   +++-w clear-at?
|     |     yang:date-and-time
++-ro output
|   ++-ro clear-finished-at?
|     yang:date-and-time
++-ro adj-rib-out-pre
++-ro routes
|   ++-ro route* [prefix path-id]
|     ++-ro prefix
|     |   inet:ipv6-prefix
|     ++-ro path-id uint32
|     ++-ro attr-index? leafref
|     ++-ro community-index? leafref
|     ++-ro ext-community-index? leafref
|     ++-ro last-modified?
|     |   yang:timeticks
|     ++-ro eligible-route? boolean
|     ++-ro ineligible-reason? identityref
|     ++-ro unknown-attributes
|     |   ++-ro unknown-attribute*
|     |     [attr-type]
++ro clear-routes (bt:clear-routes)?
  +---x clear
  +----w input
  |  +----w clear-at?
  |       yang:date-and-time
  +---ro output
  +---ro clear-finished-at?
       yang:date-and-time
++ro adj-rib-out-post
++ro routes
  ++ro route* [prefix path-id]
  |  ++ro prefix
  |   inet:ipv6-prefix
  |   +---ro path-id uint32
  |   +---ro attr-index? leafref
  |   +---ro community-index? leafref
  |   +---ro ext-community-index? leafref
  |   +---ro last-modified?
  |       yang:timeticks
  |   +---ro eligible-route?
  |       boolean
  |   +---ro ineligible-reason?
  |       identityref
  |   +---ro unknown-attributes
  |      ++ro unknown-attribute* [attr-type]
  |      |  ++ro optional? boolean
  |      |  ++ro transitive? boolean
  |      |  ++ro partial? boolean
  |      |  ++ro extended? boolean
  |      |  +---ro attr-type uint8
  |      |  +---ro attr-len? uint16
  |      |  +---ro attr-value? binary
  |      +---ro reject-reason? union
++ro clear-routes (bt:clear-routes)?
  +---x clear
  +----w input
  |  +----w clear-at?
  |       yang:date-and-time
  +---ro output
Appendix E. Complete policy tree diagram

Here is a complete tree diagram for the policy portion of the model.

module: ietf-bgp-policy

 augment /rt-pol:routing-policy/rt-pol:defined-sets:
  +--rw bgp-defined-sets
    +--rw community-sets
      +--rw community-set* [name]
        |  +--rw name     string
        |  +--rw member*  union
    +--rw ext-community-sets
      +--rw ext-community-set* [name]
        +--rw name     string
        +--rw member*  union
    +--rw large-community-sets
      +--rw large-community-set* [name]
        +--rw name     string
        +--rw member*  union
    +--rw as-path-sets
      +--rw as-path-set* [name]
        +--rw name     string
        +--rw member*  string
    +--rw next-hop-sets
      +--rw next-hop-set* [name]
        +--rw name     string
        +--rw next-hop* bgp-next-hop-type

 augment /rt-pol:routing-policy/rt-pol:policy-definitions
  /rt-pol:policy-definition/rt-pol:statements
  /rt-pol:statement/rt-pol:conditions:

  +--rw bgp-conditions
    +--rw med-eq?          uint32
    +--rw origin-eq?        bt:bgp-origin-attr-type
    +--rw next-hop-in-eq*   inet:ip-address-no-zone
    +--rw afi-safi-in*      identityref
    +--rw local-pref-eq?    uint32
    +--rw neighbor-eq*     inet:ip-address
    +--rw route-type?       enumeration
    +--rw community-count
      +--rw community-count? uint32
      +--rw (operation)?
        +--:(eq)
        |  +--rw eq?     empty
        +--:(lt-or-eq)
|   | ++--rw lt-or-eq?   empty
|   |   +--:(gt-or-eq)
|   |     +--rw gt-or-eq?   empty
|   |   ++--rw as-path-length
|   |     ++--rw as-path-length?   uint32
|   |     ++--rw (operation)?
|   |       +--:(eq)
|   |           |   ++--rw eq?         empty
|   |           |   ++--:(lt-or-eq)
|   |           |       |   ++--rw lt-or-eq?   empty
|   |           |       |   ++--:(gt-or-eq)
|       |           |       |     ++--rw gt-or-eq?   empty
|   |   ++--rw match-community-set
|   |     ++--rw community-set?       leafref
|   |     ++--rw match-set-options?   match-set-options-type
|   |   ++--rw match-ext-community-set
|   |     ++--rw ext-community-set?   leafref
|   |     ++--rw match-set-options?   match-set-options-type
|   |   ++--rw match-large-community-set
|   |     ++--rw ext-community-set?   leafref
|   |     ++--rw match-set-options?   match-set-options-type
|   |   ++--rw match-as-path-set
|   |     ++--rw as-path-set?       leafref
|   |     ++--rw match-set-options?   match-set-options-type
|   |   ++--rw match-next-hop-set
|       |   ++--rw next-hop-set?        leafref
|       |   ++--rw match-set-options?   match-set-options-type
|   | augment /rt-pol:routing-policy/rt-pol:policy-definitions
|       | /rt-pol:policy-definition/rt-pol:statements
|       | /rt-pol:statement/rt-pol:actions:
|   |   ++--rw bgp-actions
|   |     ++--rw set-route-origin?   bt:bgp-origin-attr-type
|   |     ++--rw set-local-pref?     uint32
|   |     ++--rw set-next-hop?      bgp-next-hop-type
|   |     ++--rw set-med?           bgp-set-med-type
|   |     ++--rw set-as-path-prepend
|       |         |   ++--rw repeat-n?   uint8
|   |   ++--rw set-community
|   |     ++--rw options?
|       |       |       |   bgp-set-community-option-type
|       |       |   ++--rw (method)?
|       |       |       |   +--:(inline)
|       |       |       |       |   ++--rw communities*   union
|       |       |       |       |   +--:(reference)
|       |       |       |       |     ++--rw community-set-ref?   leafref
|   |   ++--rw set-ext-community
|       |   ++--rw options?
|       |       |       |   bgp-set-community-option-type
Authors’ Addresses

Mahesh Jethanandani
Kloud Services
Email: mjethanandani@gmail.com

Keyur Patel
Arrcus
CA
United States of America
Email: keyur@arrcus.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
United States of America
Email: shares@ndzh.com

Jeffrey Haas
Juniper Networks
Email: jhaas@pfrc.org
BGP SR Policy Extensions to Enable IFIT  
draft-ietf-idr-sr-policy-ifit-03

Abstract

Segment Routing (SR) policy is a set of candidate SR paths consisting of one or more segment lists and necessary path attributes. It enables instantiation of an ordered list of segments with a specific intent for traffic steering. In-situ Flow Information Telemetry (IFIT) refers to network OAM data plane on-path telemetry techniques, in particular the most popular are In-situ OAM (IOAM) and Alternate Marking. This document defines extensions to BGP to distribute SR policies carrying IFIT information. So that IFIT methods can be enabled automatically when the SR policy is applied.

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 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."
1. Introduction

Segment Routing (SR) policy [I-D.ietf-spring-segment-routing-policy] is a set of candidate SR paths consisting of one or more segment lists and necessary path attributes. It enables instantiation of an ordered list of segments with a specific intent for traffic steering.

In-situ Flow Information Telemetry (IFIT) denotes a family of flow-oriented on-path telemetry techniques (e.g. IOAM, Alternate...
Marking), which can provide high-precision flow insight and real-time network issue notification (e.g., jitter, latency, packet loss). In particular, IFIT refers to network OAM (Operations, Administration, and Maintenance) data plane on-path telemetry techniques, including In-situ OAM (IOAM) [I-D.ietf-ippm-ioam-data] and Alternate Marking [RFC8321]. It can provide flow information on the entire forwarding path on a per-packet basis in real time.

An automatic network requires the Service Level Agreement (SLA) monitoring on the deployed service. So that the system can quickly detect the SLA violation or the performance degradation, hence to change the service deployment. For this reason, the SR policy native IFIT can facilitate the closed loop control and enable the automation of SR service.

This document defines extensions to Border Gateway Protocol (BGP) to distribute SR policies carrying IFIT information. So that IFIT behavior can be enabled automatically when the SR policy is applied.

This BGP extension allows to signal the IFIT capabilities together with the SR-policy. In this way IFIT methods are automatically activated and running. The flexibility and dynamicity of the IFIT applications are given by the use of additional functions on the controller and on the network nodes, but this is out of scope here.

IFIT is a solution focusing on network domains according to [RFC8799] that introduces the concept of specific domain solutions. A network domain consists of a set of network devices or entities within a single administration. As mentioned in [RFC8799], for a number of reasons, such as policies, options supported, style of network management and security requirements, it is suggested to limit applications including the emerging IFIT techniques to a controlled domain. Hence, the IFIT methods MUST be typically deployed in such controlled domains.

2. Motivation

IFIT Methods are being introduced in multiple protocols and below is a proper picture of the relevant documents for Segment Routing. Indeed the IFIT methods are becoming mature for Segment Routing over the MPLS data plane (SR-MPLS) and Segment Routing over IPv6 data plane (SRv6), that is the main focus of this draft:

Alternate Marking: the reference documents for the data plane are
[I-D.ietf-6man-ipv6-alt-mark] for SRv6 and
[I-D.ietf-mpls-rfc6374-sfl], [I-D.gandhi-mpls-rfc6374-sr] for SR-MPLS.

The definition of these data plane IFIT methods for SR-MPLS and SRv6
imply requirements for various routing protocols, such as BGP, and
this document aims to define BGP extensions to distribute SR policies
carrying IFIT information. This allows to signal the IFIT
capabilities so IFIT methods are automatically configured and ready
to run when the SR Policy candidate paths are distributed through
BGP.

It is to be noted that, for PCEP (Path Computation Element
Communication Protocol), [I-D.chen-pce-pcep-ifit] proposes the
extensions to PCEP to distribute paths carrying IFIT information and
therefore to enable IFIT methods for SR policy too.

3. IFIT methods for SR Policy

In-situ Operations, Administration, and Maintenance (IOAM)
[I-D.ietf-ippm-ioam-data] records operational and telemetry
information in the packet while the packet traverses a path between
two points in the network. In terms of the classification given in
RFC 7799 [RFC7799] IOAM could be categorized as Hybrid Type 1. IOAM
mechanisms can be leveraged where active OAM do not apply or do not
offer the desired results. When SR policy enables the IOAM, the IOAM
header will be inserted into every packet of the traffic that is
steered into the SR paths.

The Alternate Marking [RFC8321] technique is an hybrid performance
measurement method, per RFC 7799 [RFC7799] classification of
measurement methods. Because this method is based on marking
consecutive batches of packets. It can be used to measure packet
loss, latency, and jitter on live traffic.

This document aims to define the control plane. While the relevant
documents for the data plane application of IOAM and Alternate
Marking are respectively [I-D.ietf-ippm-ioam-ipv6-options] and
[I-D.ietf-6man-ipv6-alt-mark] for Segment Routing over IPv6 data
plane (SRv6), [I-D.ietf-mpls-rfc6374-sfl],
[I-D.gandhi-mpls-rfc6374-sr] and [I-D.gandhi-mpls-ioam-sr] for
Segment Routing over the MPLS data plane (SR-MPLS).
4. IFIT Attributes in SR Policy

As defined in [I-D.ietf-idr-segment-routing-te-policy], a new SAFI is defined (the SR Policy SAFI with codepoint 73) as well as a new NLRI. The NLRI contains the SR Policy candidate path and, according to [I-D.ietf-idr-segment-routing-te-policy], the content of the SR Policy Candidate Path is encoded in the Tunnel Encapsulation Attribute defined in [I-D.ietf-idr-tunnel-encaps] using a new Tunnel-Type called SR Policy Type with codepoint 15. 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
  SRv6 Binding SID
  Preference
  Priority
  Policy Name
  Policy Candidate Path Name
  Explicit NULL Label Policy (ENLP)
  Segment List
    Weight
    Segment
    Segment
    ...

A candidate path includes multiple SR paths, each of which is specified by a segment list. IFIT can be applied to the candidate path, so that all the SR paths can be monitored in the same way. The 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
  - SRv6 Binding SID
  - Preference
  - Priority
  - Policy Name
  - Policy Candidate Path Name
  - Explicit NULL Label Policy (ENLP)
  - IFIT Attributes
    - Segment List
      - Weight
      - Segment
      - Segment
      ...
      ...

IFIT attributes can be attached at the candidate path level as sub-TLVs. There may be different IFIT tools. The following sections will describe the requirement and usage of different IFIT tools, and define the corresponding sub-TLV encoding in BGP.

Once the IFIT attributes are signalled, if a packet arrives at the headend and, based on the types of steering described in [I-D.ietf-spring-segment-routing-policy], it may get steered into an SR Policy where IFIT methods are applied. Therefore it will be managed consequently with the corresponding IOAM or Alternate Marking information according to the enabled IFIT methods.

Note that the IFIT attributes here described can also be generalized and included as sub-TLVs for other SAFIs and NLRIs.

5. IFIT Attributes Sub-TLV

The format of the IFIT Attributes Sub-TLV is defined as follows:
Where:

Type: to be assigned by IANA.

Length: the total length of the value field not including Type and Length fields.

sub-TLVs currently defined:

* IOAM Pre-allocated Trace Option Sub-TLV,
* IOAM Incremental Trace Option Sub-TLV,
* IOAM Directly Export Option Sub-TLV,
* IOAM Edge-to-Edge Option Sub-TLV,
* Enhanced Alternate Marking (EAM) sub-TLV.

The presence of the IFIT Attributes Sub-TLV implies support of IFIT methods (IOAM and/or Alternate Marking). It is worth mentioning that IOAM and Alternate Marking can be activated one at a time or can coexist; so it is possible to have only IOAM or only Alternate Marking enabled as Sub-TLVs. The sub-TLVs currently defined for IOAM and Alternate Marking are detailed in the next sections.

In case of empty IFIT Attributes Sub-TLV, i.e. no further IFIT sub-TLV and Length=0, IFIT methods will not be activated. If two conflicting IOAM sub-TLVs are present (e.g. Pre-allocated Trace Option and Incremental Trace Option) it means that they are not usable and none of the two methods will be activated. The same applies if there is more than one instance of the sub-TLV of the same type. Anyway the validation of the individual fields of the IFIT Attributes sub-TLVs are handled by the SRPM (SR Policy Module).
The process of stopping IFIT methods can be done by setting empty IFIT Attributes Sub-TLV, while, for modifying IFIT methods parameters, the IFIT Attributes Sub-TLVs can be updated accordingly. Additionally the backward compatibility is guaranteed, since an implementation that does not understand IFIT Attributes Sub-TLV can simply ignore it.

5.1. IOAM Pre-allocated Trace Option Sub-TLV

The IOAM tracing data is expected to be collected at every node that a packet traverses to ensure visibility into the entire path a packet takes within an IOAM domain. The preallocated tracing option will create pre-allocated space for each node to populate its information.

The format of IOAM pre-allocated trace option sub-TLV is defined as follows:

```
+---------------+---------------+-------------------------------+
|    Type=1     |   Length=6    |    Namespace ID               |
|---------------+---------------+--------------+--------+-------|
|         IOAM Trace Type                      | Flags  | Rsvd  |
|----------------------------------------------|--------+-------|
```

Fig. 2 IOAM Pre-allocated Trace Option Sub-TLV

Where:

Type: 1 (to be assigned by IANA).

Length: 6, it is the total length of the value field (not including Type and Length fields).

Namespace ID: A 16-bit identifier of an IOAM-Namespace. The definition is the same as described in section 4.4 of [I-D.ietf-ippm-ioam-data].

IOAM Trace Type: A 24-bit identifier which specifies which data types are used in the node data list. The definition is the same as described in section 4.4 of [I-D.ietf-ippm-ioam-data].

Flags: A 4-bit field. The definition is the same as described in [I-D.ietf-ippm-ioam-flags] and section 4.4 of [I-D.ietf-ippm-ioam-data].

Rsvd: A 4-bit field reserved for further usage. It MUST be zero and ignored on receipt.
5.2. IOAM Incremental Trace Option Sub-TLV

The incremental tracing option contains a variable node data fields where each node allocates and pushes its node data immediately following the option header.

The format of IOAM incremental trace option sub-TLV is defined 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=2     |   Length=6    |    Namespace ID               |
|---------------+---------------+--------------+--------+-------|
|         IOAM Trace Type                      | Flags  | Rsvd  |
+----------------------------------------------+--------+-------+
```

Fig. 3 IOAM Incremental Trace Option Sub-TLV

Where:

Type: 2 (to be assigned by IANA).

Length: 6, it is the total length of the value field (not including Type and Length fields).

All the other fields definition is the same as the pre-allocated trace option sub-TLV in section 4.1.

5.3. IOAM Directly Export Option Sub-TLV

IOAM directly export option is used as a trigger for IOAM data to be directly exported to a collector without being pushed into in-flight data packets.

The format of IOAM directly export option sub-TLV is defined as follows:
Fig. 4 IOAM Directly Export Option Sub-TLV

Where:

Type: 3 (to be assigned by IANA).

Length: 12, it is the total length of the value field (not including Type and Length fields).

Namespace ID: A 16-bit identifier of an IOAM-Namespace. The definition is the same as described in section 4.4 of [I-D.ietf-ippm-ioam-data].

Flags: A 16-bit field. The definition is the same as described in section 3.2 of [I-D.ietf-ippm-ioam-direct-export].

IOAM Trace Type: A 24-bit identifier which specifies which data types are used in the node data list. The definition is the same as described in section 4.4 of [I-D.ietf-ippm-ioam-data].

Rsvd: A 4-bit field reserved for further usage. It MUST be zero and ignored on receipt.

Flow ID: A 32-bit flow identifier. The definition is the same as described in section 3.2 of [I-D.ietf-ippm-ioam-direct-export].

5.4. IOAM Edge-to-Edge Option Sub-TLV

The IOAM edge to edge option is to carry data that is added by the IOAM encapsulating node and interpreted by IOAM decapsulating node.

The format of IOAM edge-to-edge option sub-TLV is defined as follows:
Where:

Type: 4 (to be assigned by IANA).

Length: 4, it is the total length of the value field (not including Type and Length fields).

Namespace ID: A 16-bit identifier of an IOAM-Namespace. The definition is the same as described in section 4.6 of [I-D.ietf-ippm-ioam-data].

IOAM E2E Type: A 16-bit identifier which specifies which data types are used in the E2E option data. The definition is the same as described in section 4.6 of [I-D.ietf-ippm-ioam-data].

5.5. Enhanced Alternate Marking (EAM) sub-TLV

The format of Enhanced Alternate Marking (EAM) sub-TLV is defined as follows:

Where:

Type: 5 (to be assigned by IANA).

Length: 4, it is the total length of the value field (not including Type and Length fields).
FlowMonID: A 20-bit identifier to uniquely identify a monitored flow within the measurement domain. The definition is the same as described in section 5.3 of [I-D.ietf-6man-ipv6-alt-mark].

Period: Time interval between two alternate marking period. The unit is second.

H: A flag indicating that the measurement is Hop-By-Hop.

E: A flag indicating that the measurement is end to end.

R: A 2-bit field reserved for further usage. It MUST be zero and ignored on receipt.

6. SR Policy Operations with IFIT Attributes

The details of SR Policy installation and use are specified in [I-D.ietf-spring-segment-routing-policy]. This document complements SR Policy Operations described in [I-D.ietf-idr-segment-routing-te-policy] by adding the IFIT Attributes.

The operations described in [I-D.ietf-idr-segment-routing-te-policy] are always valid. The only difference is the addition of IFIT Attributes Sub-TLVs for the SR Policy NLRI, that can affect its acceptance by a BGP speaker, but the implementation MAY provide an option for ignoring the unrecognized or unsupported IFIT sub-TLVs. SR Policy NLRIs that have been determined acceptable, usable and valid can be evaluated for propagation, including the IFIT information.

The error handling actions are also described in [I-D.ietf-idr-segment-routing-te-policy], indeed A BGP Speaker MUST perform the syntactic validation of the SR Policy NLRI to determine if it is malformed, including the TLVs/sub-TLVs. In case of any error detected, either at the attribute or its TLV/sub-TLV level, the "treat-as-withdraw" strategy MUST be applied.

The validation of the IFIT Attributes sub-TLVs introduced in this document MUST be performed to determine if they are malformed or invalid. The validation of the individual fields of the IFIT Attributes sub-TLVs are handled by the SRPM (SR Policy Module).

7. IANA Considerations

This document defines a new sub-TLV in the registry "BGP Tunnel Encapsulation Attribute sub-TLVs" to be assigned by IANA:
This document requests creation of a new registry called "IFIT Attributes Sub-TLVs". The allocation policy of this registry is "Specification Required" according to RFC 8126 [RFC8126].

The following initial Sub-TLV codepoints are assigned by this document:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IOAM Pre-allocated Trace Option Sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>2</td>
<td>IOAM Incremental Trace Option Sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>3</td>
<td>IOAM Directly Export Option Sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>4</td>
<td>IOAM Edge-to-Edge Option Sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>5</td>
<td>Enhanced Alternate Marking Sub-TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>

8. Security Considerations

The security mechanisms of the base BGP security model apply to the extensions described in this document as well. See the Security Considerations section of [I-D.ietf-idr-segment-routing-te-policy].

SR operates within a trusted SR domain RFC 8402 [RFC8402] and its security considerations also apply to BGP sessions when carrying SR Policy information. The isolation of BGP SR Policy SAFI peering sessions may be used to ensure that the SR Policy information is not advertised outside the SR domain. Additionally, only trusted nodes (that include both routers and controller applications) within the SR domain must be configured to receive such information.

Implementation of IFIT methods (IOAM and Alternate Marking) are mindful of security and privacy concerns, as explained in [I-D.ietf-ippm-ioam-data] and RFC 8321 [RFC8321]. Anyway incorrect IFIT parameters in the BGP extension SHOULD NOT have an adverse effect on the SR Policy as well as on the network, since it affects only the operation of the telemetry methodology.

IFIT data MUST be propagated in a limited domain in order to avoid malicious attacks and solutions to ensure this requirement are
respectively discussed in [I-D.ietf-ippm-ioam-data] and
[I-D.ietf-6man-ipv6-alt-mark].

IFIT methods (IOAM and Alternate Marking) are applied within a
derived domain where the network nodes are locally administered.
A limited administrative domain provides the network administrator
with the means to select, monitor and control the access to the
network, making it a trusted domain also for the BGP extensions
defined in this document.

9. Acknowledgements

The authors of this document would like to thank Ketan Talaulikar,
Joel Halpern, Jie Dong for their comments and review of this
document.

10. References

10.1. Normative References

[I-D.ietf-6man-ipv6-alt-mark]
Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R.
Pang, "IPv6 Application of the Alternate Marking Method",
draft-ietf-6man-ipv6-alt-mark-12 (work in progress),
October 2021.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P.,
Jain, D., and S. Lin, "Advertising Segment Routing
Policies in BGP", draft-ietf-idr-segment-routing-te-
policy-14 (work in progress), November 2021.

[I-D.ietf-idr-tunnel-encaps]
Patel, K., Veld, G. V. D., Sangli, S. R., and J. Scudder,
"The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-
tunnel-encaps-22 (work in progress), January 2021.

[I-D.ietf-ippm-ioam-data]
Brockners, F., Bhandari, S., and T. Mizrahi, "Data Fields
for In-situ OAM", draft-ietf-ippm-ioam-data-17 (work in
progress), December 2021.

[I-D.ietf-ippm-ioam-direct-export]
Song, H., Gafni, B., Zhou, T., Li, Z., Brockners, F.,
Bhandari, S., Sivakolundu, R., and T. Mizrahi, "In-situ
OAM Direct Exporting", draft-ietf-ippm-ioam-direct-
export-07 (work in progress), October 2021.


10.2. Informative References

[I-D.chen-pce-pcep-ifit]

[I-D.gandhi-mpls-ioam-sr]

[I-D.gandhi-mpls/rfc6374-sr]

[I-D.ietf-mpls-rfc6374-sfl]

Appendix A.

Authors’ Addresses

Fengwei Qin
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing
China

Email: qinfengwei@chinamobile.com
Traffic Steering using BGP Flowspec with SRv6 Policy
draft-jiang-idr-ts-flowspec-srv6-policy-07

Abstract

BGP Flow Specification (FlowSpec) [RFC8955] [RFC8956] has been proposed to distribute BGP FlowSpec NLRI to FlowSpec clients to mitigate (distributed) denial-of-service attacks, and to provide traffic filtering in the context of a BGP/MPLS VPN service. Recently, traffic steering applications in the context of SRv6 using FlowSpec also attract attention. This document introduces the usage of BGP FlowSpec to steer packets into an SRv6 Policy.

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 September 24, 2022.
1. Introduction

Segment Routing IPv6 (SRv6) is a protocol designed to forward IPv6 data packets on a network using the source routing model. SRv6 enables the ingress to add a segment routing header (SRH) [RFC8754] to an IPv6 packet and push an explicit IPv6 address stack into the SRH. After receiving the packet, each transit node updates the IPv6 destination IP address in the packet and segment list to implement hop-by-hop forwarding.

SRv6 Policy [I-D.ietf-spring-segment-routing-policy] is a tunneling technology developed based on SRv6. An SRv6 Policy is a set of candidate paths consisting of one or more segment lists, that is, segment ID (SID) lists. Each SID list identifies an end-to-end path from the source to the destination, instructing a device to forward
traffic through the path rather than the shortest path computed using an IGP. The header of a packet steered into an SRv6 Policy is augmented with an ordered list of segments associated with that SRv6 Policy, so that other devices on the network can execute the instructions encapsulated into the list.

The headend of an SRv6 Policy may learn multiple candidate paths for an SRv6 Policy. Candidate paths may be learned via a number of different mechanisms, e.g., CLI, NetConf, PCEP, or BGP.

[RFC8955] [RFC8956] defines the flow specification (FlowSpec) that allows to convey flow specifications and traffic Action/Rules associated (rate-limiting, redirect, remark ...). BGP Flow specifications are encoded within the MP_REACH_NLRI and MP_UNREACH_NLRI attributes. Rules (Actions associated) are encoded in Extended Community attribute. The BGP Flow Specification function allows BGP Flow Specification routes that carry traffic policies to be transmitted to BGP Flow Specification peers to steer traffic.

This document proposes BGP flow specification usage that are used to steer data flow into an SRv6 Policy as well as to indicate Tailend function.

2. Definitions and Acronyms

   o FlowSpec: Flow Specification
   o SR: Segment Routing
   o SRv6: IPv6 Segment Routing
   o SID: Segment Identifier
   o SRH: Segment Routing Header
   o TE: Traffic Engineering

3. Operations

An SRv6 Policy [I-D.ietf-spring-segment-routing-policy] is identified through the tuple <headend, color, endpoint>. In the context of a specific headend, one may identify an SRv6 policy by the <color, endpoint> tuple. The headend is the node where the SRv6 policy is instantiated/implemented. The headend is specified as an IPv4 or IPv6 address and is expected to be unique in the domain. The endpoint indicates the destination of the SRv6 policy. The endpoint is specified as an IPv6 address and is expected to be unique in the domain. The color is a 32-bit numerical value that associates the
SRv6 Policy, and it defines an application-level network Service Level Agreement (SLA) policy.

Assume one or multiple SRv6 Policies are already setup in the SRv6 HeadEnd device. In order to steer traffic into a specific SRv6 policy at the Headend, one can use the SRv6 color extended community and endpoint to map to a satisfying SRv6 policy, and steer traffic into this specific policy.

[I-D.ietf-idr-flowspec-redirect-ip] defines the redirect to IPv4 and IPv6 Next-hop action. The IPv6 next-hop address in the Flow-spec Redirect to IPv6 Extended Community can be used to specify the endpoint of the SRv6 Policy. When the packets reach to the TailEnd device, some specific function information identifiers can be used to decide how to further process the flows. Several endpoint functions are already defined, e.g., End.DT6: Endpoint with decapsulation and IPv6 table lookup, and End.DX6: Endpoint with decapsulation and IPv6 cross-connect. The BGP Prefix-SID defined in [RFC8669] is utilized to enable SRv6 VPN services [I-D.ietf-bess-srv6-services]. SRv6 Services TLVs within the BGP Prefix-SID Attribute can be used to indicate the endpoint functions.

This document proposes to carry the Color Extended Community and BGP Prefix-SID Attribute in the context of a Flowspec NLRI [RFC8955] [RFC8956] to an SRv6 Headend to steer traffic into one SRv6 policy, as well as to indicate specific Tailend functions.

In this document, the usage of at most one Color Extended Community in combination at most one BGP Prefix SID Attribute is discussed. For the case that a flowspec route carries multiple Color Extended Communities and/or a BGP Prefix SID Attribute, a protocol extension to Flowspec is required, and is thus out of the scope of this document.

However, the method proposed in this document still supports load balancing to the tailend device. To achieve that, the headend device can set up multiple paths in one SRv6 policy, and use a Flowspec route to indicate the specific SRv6 policy.

4. Application Example

In following scenario, BGP FlowSpec Controller signals the filter rules, the redirect action, the policy color and the function information (SRv6 SID: Service_id_x) to the HeadEnd device.
Flowspec route to HeadEnd:
   NLRI: Filter Rules
   Redirect to IPv6 Nexthop: TailEnd’s Address
   Policy Color: C1
   PrefixSID: Service_id_x
       .-----.
       (   )
       V  .--(       )--.

  +-------+  (                 )  +-------+
  |       |_( SRv6 Core Network )_|       |
  |HeadEnd ( ================> ) |TailEnd|
  +-------+  (SR List<S1, S2, S3>)  +-------+
            '--(             )--'
            (       )
            (e.g.: End.DT4 or End.DT6 or others)
            '-----'

Figure 1: Steering the Flow into SRv6 Policy (Option 1)

When the HeadEnd device (as a Flowspec client) receives such instructions, it will steer the flows matching the criteria in the Flowspec route into the SRv6 Policy matching the tuple (Endpoint: TailEnd’s Address, Color: C1). And the packets of such flows will be encapsulated with SRH using the SR List<S1, S2, S3, Service_id_x>. When the packets reach to the TailEnd device, they will be further processed per the function denoted by the Service_id_x.

When the HeadEnd device determines (with the help of SRv6 SID Structure) that the Service SID belongs to the same SRv6 Locator as the last SRv6 SID of the TailEnd device in the SRv6 Policy segment list, it MAY exclude that last SRv6 SID when steering the service flow. For example, the effective segment list of the SRv6 Policy associated with SID list <S1, S2, S3> would be <S1, S2, Service_id_x>.

If the last SRv6 SID (For example, S3 we use here) of the TailEnd device in the SRv6 Policy segment list is USD-flavored, an SRv6 Service SID (e.g., End.DT4 or End.DT6) is not required when BGP FlowSpec Controller send the Flowspec route to the HeadEnd device (as a Flowspec client).
When the HeadEnd device (as a Flowspec client) receives such instructions, it will steer the flows matching the criteria in the Flowspec route into the SRv6 Policy matching the tuple (Endpoint: TailEnd’s Address, Color: C2). And the packets of such flows will be encapsulated with SRH using the SR List<S1, S2, S3>. When the packets reach to the TailEnd device, they will be further processed per the function denoted by the USD-flavored SRv6 SID S3.

At this point, the work discusses the matching of global routing table prefixes.

For the cases of intra-AS and inter-AS traffic steering using this method, the usages of Flowspec Color Extended Community with BGP prefix SID are the same for both scenarios. The difference lies between the local SRv6 policy configurations. For the inter-domain case, the operator can configure an inter-domain SRv6 policy/path at the Headend device. For the intra-domain case, the operator can configure an intra-domain SRv6 policy/path at the Headend device.
5. Running Code

5.1. Interop-test Status

The Traffic Steering using BGP Flowspec with SRv6 Policy mechanism has been implemented on the following hardware devices, software implementations and SDN controllers. They had also successfully participated in the series of joint interoperability testing events hosted by China Mobile from July 2021 to October 2021. The following hardware devices and software implementations had successfully passed the interoperability testing (in alphabetical order).

Routers:

<table>
<thead>
<tr>
<th>Vendors</th>
<th>Device Model</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huawei</td>
<td>NE40-X8A</td>
<td>NE40E V800R021C00SPC091T</td>
</tr>
<tr>
<td>New H3C</td>
<td>CR16010H-FA</td>
<td>Version 7.1.075, ESS 8305</td>
</tr>
<tr>
<td>Ruijie</td>
<td>RG-N8010-R</td>
<td>N8000-R_RGOS 12.8(1)B08T1</td>
</tr>
<tr>
<td>ZTE</td>
<td>M6000-8S Plus</td>
<td>V5.00.10(5.60.5)</td>
</tr>
</tbody>
</table>

Controllers:

<table>
<thead>
<tr>
<th>Vendors</th>
<th>Device Model</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>China Unitecs</td>
<td>I-T-E SC</td>
<td>V1.3.6P3</td>
</tr>
<tr>
<td>Huawei</td>
<td>NCE-IP</td>
<td>V100R021C00</td>
</tr>
<tr>
<td>Ruijie</td>
<td>RG-ONC-AIO-H</td>
<td>RG-ION-WAN-CLOUD_2.00T1</td>
</tr>
<tr>
<td>ZTE</td>
<td>ZENIC ONE</td>
<td>R22V16.21.20</td>
</tr>
</tbody>
</table>

5.2. Deployment Status

TBD

6. IANA Considerations

No IANA actions are required for this document.
7. Security Considerations

This document does not change the security properties of SRv6 and BGP.

8. Contributors

The following people made significant contributions to this document:

Yunan Gu
Huawei
Email: guyunan@huawei.com

Haibo Wang
Huawei
Email: rainsword.wang@huawei.com

Jie Dong
Huawei
Email: jie.dong@huawei.com

Xue Yang
China Mobile
Email: yangxuewl@chinamobile.com

9. Acknowledgements

The authors would like to acknowledge the review and inputs from Jeffrey Haas, Kaliraj Vairavakkalai, Robin Li, Acee Lindem, Gunter Van De Velde, John Scudder, Rainbow Wu and Gang Yang.

10. References

10.1. Normative References

[I-D.ietf-bess-srv6-services]

[I-D.ietf-idr-flowspec-redirect-ip]
10.2. Informative References


Authors’ Addresses

Wenying Jiang
China Mobile
Beijing
China

Email: jiangwenying@chinamobile.com

Yisong Liu
China Mobile
Beijing
China

Email: liuyisong@chinamobile.com

Shuanglong Chen
Huawei
Beijing
China

Email: chenshuanglong@huawei.com

Shunwan Zhuang
Huawei
Beijing
China

Email: zhuangshunwan@huawei.com
BGP Classful Transport Planes
draft-kaliraj-idr-bgp-classful-transport-planes-14

Abstract

This document specifies a mechanism, referred to as "service mapping", to express association of overlay routes with underlay routes satisfying a certain SLA, using BGP. The document describes a framework for classifying underlay routes into transport classes, and mapping service routes to specific transport class.

The "Transport class" construct maps to a desired SLA, and can be used to realize the "Topology Slice" in 5G Network slicing architecture.

This document specifies BGP protocol procedures that enable dissemination of such service mapping information that may span multiple co-operating administrative domains. These domains may be administrated by the same provider or closely co-ordinating provider networks.

It makes it possible to advertise multiple tunnels to the same destination address, thus avoiding need of multiple loopbacks on the egress node.

A new BGP transport layer address family (SAFI 76) is defined for this purpose that uses RFC-4364 technology and follows RFC-8277 NLRI encoding. This new address family is called "BGP Classful Transport", aka BGP CT.
It carries transport prefixes across tunnel domain boundaries (e.g. in Inter-AS Option-C networks), parallel to BGP LU (SAFI 4). It disseminates "Transport class" information for the transport prefixes across the participating domains, which is not possible with BGP LU. This makes the end-to-end network a "Transport Class" aware tunneled network.

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 28 October 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

To facilitate service mapping, the tunnels in a network can be grouped by the purpose they serve into a "Transport Class". The tunnels could be created using any signaling protocol, such as LDP, RSVP, BGP LU or SPRING. The tunnels could also use native IP or IPv6, as long as they can carry MPLS payload. Tunnels may exist between different pair of end points. Multiple tunnels may exist
between the same pair of end points.

Thus, a Transport Class consists of tunnels created by various protocols that satisfy the properties of the class. For example, a "Gold" transport class may consist of tunnels that traverse the shortest path with fast re-route protection, a "Silver" transport class may hold tunnels that traverse shortest paths without protection, a "To NbrAS Foo" transport class may hold tunnels that exit to neighboring AS Foo, and so on.

The extensions specified in this document can be used to create a BGP transport tunnel that potentially spans domains, while preserving its Transport Class. Examples of domain are Autonomous System (AS), or IGP area. Within each domain, there is a second level underlay tunnel used by BGP to cross the domain. The second level underlay tunnels could be heterogeneous: Each domain may use a different type of tunnel (e.g. MPLS, IP, GRE), or use a different signaling protocol. A domain boundary is demarcated by a rewrite of BGP nexthop to 'self' while re-advertising tunnel routes in BGP. Examples of domain boundary are inter-AS links and inter-region ABRs. The path uses MPLS label-switching when crossing domain boundary and uses the native intra-AS tunnel of the desired transport class when traversing within a domain.

Overlay routes carry sufficient indication of the Transport Classes they should be encapsulated over, in form of BGP community called the "Mapping community". Based on the mapping community, "route resolution" procedure on the ingress node selects from the corresponding Transport Class an appropriate tunnel whose destination matches (LPM) the nexthop of the overlay route. If the overlay route is carried in BGP, the protocol nexthop (or, PNH) is generally carried as an attribute of the route.

The PNH of the overlay route is also referred to as "service endpoint" (SEP). The service endpoint may exist in the same domain as the service ingress node or lie in a different domain, adjacent or non-adjacent. In the former case, reachability to the SEP is provided by an intra-domain tunneling protocol, and in the latter case, reachability to the SEP is via BGP transport families.

In this architecture, the intra-domain transport protocols (e.g. RSVP, SRTE) are also "Transport Class aware", and they publish ingress routes in Transport RIB associated with the Transport Class, at the tunnel ingress node. These routes are then redistributed into BGP CT to be advertised to adjacent domains. It is outside the scope of this document how exactly the transport protocols are made transport class aware, though configuration on the tunnel ingress node is a simple mechanism to achieve it.
This document describes mechanisms to:

Model a "Transport Class" as "Transport RIB" on a router, consisting of tunnel ingress routes of a certain class.

Enable service routes to resolve over an intended Transport Class by virtue of carrying the appropriate "Mapping community". Which results in using the corresponding Transport RIB for finding next hop reachability.

Advertise tunnel ingress routes in a Transport RIB via BGP without any path hiding, using BGP VPN technology and Add-path. Such that overlay routes in the receiving domains can also resolve over tunnels of associated Transport Class.

Provide a way for co-operating domains to reconcile any differences in extended community namespaces, and interoperate between different transport signaling protocols in each domain.

In this document we focus mainly on MPLS as the intra-domain transport tunnel forwarding, but the mechanisms described here would work in similar manner for non-MPLS (e.g. IP, GRE, UDP) transport tunnel forwarding technologies too.

This document assumes MPLS forwarding when crossing domain boundaries, as that is the defacto standard in deployed networks today. But mechanisms specified in this document can also support different forwarding technologies (e.g. SRv6).

Section [SRV6-INTER-DOMAIN] in this document describes adaptation of BGP CT over SRv6 data plane.

The document Seamless Segment Routing [Seamless-SR] describes various use cases and applications of procedures described in this document.

2. Terminology

LSP: Label Switched Path.

TE : Traffic Engineering.

SN : Service Node.

BN : Border Node.

TN : Transport Node, P-router.

BGP-VPN : VPNs built using RFC4364 mechanisms.
RT : Route-Target extended community.

RD : Route-Distinguisher.

PNH : Protocol-Nexthop address carried in a BGP Update message.

SEP : Service End point, the PNH of a Service route.

LPM : Longest Prefix Match.

Service Family : BGP address family used for advertising routes for "data traffic", as opposed to tunnels.

Transport Family : BGP address family used for advertising tunnels, which are in turn used by service routes for resolution.

Transport Tunnel : A tunnel over which a service may place traffic. These tunnels can be GRE, UDP, LDP, RSVP, or SR-TE.

Tunnel Domain : A domain of the network containing SN and BN, under a single administrative control that has a tunnel between SN and BN. An end-to-end tunnel spanning several adjacent tunnel domains can be created by "stitching" them together using labels.

Transport Class : A group of transport tunnels offering the same type of service.

Transport Class RT : A Route-Target extended community used to identify a specific Transport Class.

Transport RIB : At the SN and BN, a Transport Class has an associated Transport RIB that holds its tunnel routes.

Transport Plane : An end to end plane comprising of transport tunnels belonging to same transport class. Tunnels of same transport class are stitched together by BGP route readvertise-ments with nexthop-self, to span across domain boundaries using Label-Swap forwarding mechanism similar to Inter-AS option-b.

Mapping Community : BGP Community/Extended-community on a service route, that maps it to resolve over a Transport Class.

3. Transport Class

A Transport Class is defined as a set of transport tunnels that share certain characteristics useful for underlay selection.
On the wire, a transport class is represented as the Transport Class RT, which is a new Route-Target extended community.

A Transport Class is configured at SN and BN, along with attributes like RD and Route-Target. Creation of a Transport Class instantiates the associated Transport RIB and a Transport routing instance to contain them all.

The operator may configure a SN/BN to classify a tunnel into an appropriate Transport Class, which causes the tunnel’s ingress routes to be installed in the corresponding Transport RIB. At a BN, these tunnel routes may then be advertised into BGP CT.

Alternatively, a router receiving the transport routes in BGP with appropriate signaling information can associate those ingress routes to the appropriate Transport Class. E.g. for Classful Transport family (SAFI 76) routes, the Transport Class RT indicates the Transport Class. For BGP LU family(SAFI 4) routes, import processing based on Communities or inter-AS source-peer may be used to place the route in the desired Transport Class.

When the ingress route is received via SRTE [SRTE], which encodes the Transport Class as an integer 'Color' in the NLRI as "Color:Endpoint", the 'Color' is mapped to a Transport Class during import processing. SRTE ingress route for 'Endpoint' is installed in that transport class. The SRTE route when advertised out to BGP speakers will then be advertised in Classful Transport family with Transport Class RT and a new label. The MPLS swap route thus installed for the new label will pop the label and deliver decapsulated traffic into the path determined by SRTE route.

RFC8664 [RFC8664] extends PCEP to carry SRTE Color. This color association thus learnt is also mapped to a Transport Class thus associating the PCEP signaled SRTE LSP with the desired Transport Class.

Similarly, PCEP-RSVP-COLOR [PCEP-RSVP-COLOR] extends PCEP to carry RSVP Color. This color association thus learnt is also mapped to a Transport Class thus associating the PCEP signaled RSVP LSP with the desired Transport Class.

4. "Transport Class" Route Target Extended Community

This document defines a new type of Route Target, called "Transport Class" Route Target Extended Community.
"Transport Class" Route Target extended community is a transitive extended community EXT-COMM [RFC4360] of extended-type, with a new Format (Type high = 0xa) and SubType as 0x2 (Route Target).

This new Route Target Format has the following encoding:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type= 0xa</td>
<td>SubType= 0x02</td>
<td>Reserved</td>
<td>Transport Class ID</td>
</tr>
<tr>
<td>+-------------------------------+-------------------------------+----------------+------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+-------------------------------+-------------------------------+----------------+------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

"Transport Class" Route Target Extended Community

Type: 1 octet

Type field contains value 0xa.

SubType: 1 octet

Subtype field contain 0x2. This indicates 'Route Target'.

Transport Class ID: 4 octets

The least significant 32-bits of the value field contain the "Transport Class" identifier, which is a 32-bit integer.

The remaining 2 octets after SubType field are Reserved, they MUST be set to zero by originator, and ignored, left unaltered by receiver.

The "Transport class" Route Target Extended community follows the mechanisms for VPN route import, export as specified in BGP-VPN [RFC4364], and follows the Route Target Contrain mechanisms as specified in VPN-RTC [RFC4684]

A BGP speaker that implements RT Constraint VPN-RTC [RFC4684] MUST apply the RT Constraint procedures to the "Transport class" Route Target Extended community as-well.

The Transport Class Route Target Extended community is carried on Classful Transport family routes, and allows associating them with appropriate Transport RIBs at receiving BGP speakers.
Use of the Transport Class Route Target Extended community with a new Type code avoids conflicts with any VPN Route Target assignments already in use for service families.

5. Transport RIB

A Transport RIB is a routing-only RIB that is not installed in forwarding path. However, the routes in this RIB are used to resolve reachability of overlay routes’ PNH. Transport RIB is created when the Transport Class it represents is configured.

Overlay routes that want to use a specific Transport Class confine the scope of nexthop resolution to the set of routes contained in the corresponding Transport RIB. This Transport RIB is the "Routing Table" referred in Section 9.1.2.1 RFC4271 (https://www.rfc-editor.org/rfc/rfc4271#section-9.1.2.1)

Routes in a Transport RIB are exported out in 'Classful Transport' address family.

6. Transport Routing Instance

A BGP VPN routing instance that is a container for the Transport RIB. It imports, and exports routes in this RIB with Transport Class RT. Tunnel destination addresses in this routing instance’s context come from the "provider namespace". This is different from user VRFs for e.g., which contain prefixes in "customer namespace"

The Transport Routing instance uses the RD and RT configured for the Transport Class.

7. Nexthop Resolution Scheme

An implementation may provide an option for the service route to resolve over less preferred Transport Classes, should the resolution over preferred, or "primary" Transport Class fail.

To accomplish this, the set of service routes may be associated with a user-configured "resolution scheme", which consists of the primary Transport Class, and optionally, an ordered list of fallback Transport Classes.

A community called as "Mapping Community" is configured for a "resolution scheme". A Mapping community maps to exactly one resolution scheme. A resolution scheme comprises of one primary transport class and optionally one or more fallback transport classes.
A BGP route is associated with a resolution scheme during import processing. The first community on the route that matches a mapping community of a locally configured resolution scheme is considered the effective mapping community for the route. The resolution scheme thus found is used when resolving the route’s PNH. If a route contains more than one mapping community, it indicates that the route considers these multiple mapping communities as equivalent. So the first community that maps to a resolution scheme is chosen.

A transport route received in BGP Classful Transport family SHOULD use a resolution scheme that contains the primary Transport Class without any fallback to best effort tunnels. The primary Transport Class is identified by the Transport Class RT carried on the route. Thus Transport Class RT serves as the Mapping Community for Classful Transport routes.

A service route received in a BGP service family MAY map to a resolution scheme that contains the primary Transport Class identified by the mapping community on the route, and a fallback to best effort tunnels transport class. The primary Transport Class is identified by the Mapping community carried on the route. For e.g. the Extended Color community may serve as the Mapping Community for service routes. Color:0:<n> MAY map to a resolution scheme that has primary transport class <n>, and a fallback to best-effort transport class.

8. BGP Classful Transport Family NLRI

The Classful Transport (CT) family will use the existing AFI of IPv4 or IPv6, and a new SAFI 76 "Classful Transport" that will apply to both IPv4 and IPv6 AFIs. These AFI, SAFI pair of values MUST be negotiated in Multiprotocol Extensions capability described in [RFC4760] to be able to send and receive BGP CT routes.

The "Classful Transport" SAFI NLRI itself is encoded as specified in https://tools.ietf.org/html/rfc8277#section-2 [RFC8277].

When AFI is IPv4 the "Prefix" portion of Classful Transport family NLRI consists of an 8-byte RD followed by an IPv4 prefix. When AFI is IPv6 the "Prefix" consists of an 8-byte RD followed by an IPv6 prefix.

Attributes on a Classful Transport route include the Transport Class Route-Target extended community, which is used to leak the route into the right Transport RIBs on SNs and BNs in the network.

SAFI 76 routes can be sent with either IPv4 or IPv6 nexthop. The type of nexthop is inferred from the length of nexthop.
When the length of Next Hop Address field is 24 (or 48) the nexthop address is of type VPN-IPv6 with 8-octet RD set to zero (potentially followed by the link-local VPN-IPv6 address of the next hop with an 8-octet RD set to zero).

When the length of Next Hop Address field is 12 the nexthop address is of type VPN-IPv4 with 8-octet RD set to zero.

9. Comparison with other families using RFC-8277 encoding

SAFI 128 (Inet-VPN) is a RFC8277 encoded family that carries service prefixes in the NLRI, where the prefixes come from the customer namespaces, and are contextualized into separate user virtual service RIBs called VRFs, using RFC4364 procedures.

SAFI 4 (BGP LU) is a RFC8277 encoded family that carries transport prefixes in the NLRI, where the prefixes come from the provider namespace.

SAFI 76 (Classful Transport) is a RFC8277 encoded family that carries transport prefixes in the NLRI, where the prefixes come from the provider namespace, but are contextualized into separate Transport RIBs, using RFC4364 procedures.

It is worth noting that SAFI 128 has been used to carry transport prefixes in "L3VPN Inter-AS Carrier’s carrier" scenario, where BGP LU/LDP prefixes in CsC VRF are advertised in SAFI 128 towards the remote-end baby carrier.

In this document a new AFI/SAFI is used instead of reusing SAFI 128 to carry these transport routes, because it is operationally advantageous to segregate transport and service prefixes into separate address families, RIBs. E.g. It allows to safely enable "per-prefix" label allocation scheme for Classful Transport prefixes without affecting SAFI 128 service prefixes which may have huge scale. "per prefix" label allocation scheme keeps the routing churn local during topology changes.

A new family also facilitates having a different readvertisement path of the transport family routes in a network than the service route readvertisement path. viz. Service routes (Inet-VPN) are exchanged over an EBGP multihop session between Autonomous systems with nexthop unchanged; whereas Classful Transport routes are readvertised over EBGP single hop sessions with "nexthop-self" rewrite over inter-AS links.
The Classful Transport family is similar in vein to BGP LU, in that it carries transport prefixes. The only difference is, it also carries in Route Target an indication of which Transport Class the transport prefix belongs to, and uses RD to disambiguate multiple instances of the same transport prefix in a BGP Update.

10. Protocol Procedures

This section summarizes the procedures followed by various nodes speaking Classful Transport family

Preparing the network for deploying Classful Transport planes

Operator decides on the Transport Classes that exist in the network, and allocates a Route-Target to identify each Transport Class.

Operator configures Transport Classes on the SNs and BNs in the network with unique Route-Distinguishers and Route-Targets.

Implementations may provide automatic generation and assignment of RD, RT values for a transport routing instance; they MAY also provide a way to manually override the automatic mechanism, in order to deal with any conflicts that may arise with existing RD, RT values in the different network domains participating in a deployment.

Origination of Classful Transport route:

At the ingress node of the tunnel’s home domain, the tunneling protocols install routes in the Transport RIB associated with the Transport Class the tunnel belongs to.

The ingress node then advertises this tunnel destination into BGP as a Classful Transport family route with NLRI RD:TunnelEndpoint, attaching a ‘Transport Class’ Route Target that identifies the Transport Class. This BGP CT route is advertised to EBGP peers and IBGP peers which are RR-clients. This route MUST NOT be advertised to the IBGP peers who are not RR-clients.

Alternatively, the egress node of the tunnel i.e. the tunnel endpoint can originate the same BGP Classful Transport route, with NLRI RD:TunnelEndpoint and PNH TunnelEndpoint, which will resolve over the tunnel route at the ingress node. When the tunnel is up, the Classful Transport BGP route will become usable and get re-advertised.
Unique RD SHOULD be used by the originator of a Classful Transport route to disambiguate the multiple BGP advertisements for a transport end point.

Ingress node receiving Classful Transport route

On receiving a BGP Classful Transport route with a PNH that is not directly connected, e.g., an IBGP-route, a mapping community on the route (the Transport Class RT) indicates which Transport Class this route maps to. The routes in the associated Transport RIB are used to resolve the received PNH. If there does not exist a route in the Transport RIB matching the PNH, the Classful Transport route is considered unusable, and MUST NOT be re-advertised further.

Border node re-advertising Classful Transport route with nexthop self:

The BN allocates an MPLS label to advertise upstream in Classful Transport NLRI. The BN also installs an MPLS swap-route for that label that swaps the incoming label with a label received from the downstream BGP speaker, or pops the incoming label. And then pushes received traffic to the transport tunnel or direct interface that the Classful Transport route’s PNH resolved over.

The label SHOULD be allocated with "per-prefix" label allocation semantics. RD is stripped from the BGP CT NLRI prefix when a BGP CT route is leaked to a Transport RIB. The IP prefix in the transport RIB context (IP-prefix, Transport-Class) is used as the key to do per-prefix label allocation. This helps in avoiding BGP CT route churn through out the CT network when a failure happens in a domain. The failure is not propagated further than the BN closest to the failure.

The value of advertised MPLS label is locally significant, and is dynamic by default. The BN may provide option to allocate a value from a statically carved out range. This can be achieved using locally configured export policy, or via mechanisms described in BGP Prefix-SID [RFC8669].

Border node receiving Classful Transport route on EBGP:

If the route is received with PNH that is known to be directly connected, e.g., EBGP single-hop peering address, the directly connected interface is checked for MPLS forwarding capability. No other nexthop resolution process is performed, as the inter-AS link can be used for any Transport Class.
If the inter-AS links should honor Transport Class, then the BN
SHOULD follow procedures of an Ingress node described above, and
perform nexthop resolution process. The interface routes SHOULD
be installed in the Transport RIB belonging to the associated
Transport Class.

Avoiding path-hiding through Route Reflectors

When multiple BNs exist that advertise a RDn:PEn prefix to RRs,
the RRs may hide all but one of the BNs, unless ADDPATH [RFC7911]
exists used for the Classful Transport family. This is similar to
L3VPN option-B scenarios. Hence ADDPATH SHOULD be used for
Classful Transport family, to avoid path-hiding through RRs.

Avoiding loop between Route Reflectors in forwarding path

Pair of redundant ABRs acting as RR with nexthop-self may chose
each other as best path instead of the upstream ASBR, causing a
traffic forwarding loop.

Implementations SHOULD provide a way to alter the tie-breaking
rule specified in BGP RR [RFC4456] to tie-break on CLUSTER_LIST
step before ROUTER-ID step, when performing path selection for BGP
CT routes. RFC4456 considers pure RR which is not in forwarding
path. When RR is in forwarding path and reflects routes with
nexthop-self, which is the case for ABR BNs in a BGP transport
network, this rule may cause loops. This document suggests the
following modification to the BGP Decision Process Tie Breaking
rules (Sect. 9.1.2.2, [RFC4271]) when doing path selection for BGP
CT family routes:

The following rule SHOULD be inserted between Steps e) and f): a
BGP Speaker SHOULD prefer a route with the shorter CLUSTER_LIST
length. The CLUSTER_LIST length is zero if a route does not carry
the CLUSTER_LIST attribute.

Some deployment considerations can also help in avoiding this
problem:

- IGP metric should be assigned such that "ABR to redundant ABR"
cost is inferior than "ABR to upstream ASBR" cost.

- Tunnels belonging to special Transport classes SHOULD NOT be
  provisioned between ABR to ABRs. This will ensure that the
  route received from an ABR with nexthop-self will not be usable
  at a redundant ABR.
This avoids possibility of such loops altogether, irrespective of whether the path selection modification mentioned above is implemented.

Ingress node receiving service route with mapping community

Service routes received with mapping community resolve using Transport RIBs determined by the resolution scheme. If the resolution process does not find an usable Classful Transport route or tunnel route in any of the Transport RIBs, the service route MUST be considered unusable for forwarding purpose.

Coordinating between domains using different community namespaces.

Cooperating option-C domains may sometimes not agree on RT, RD, Mapping-community or Transport Route Target values because of differences in community namespaces; e.g. during network mergers or renumbering for expansion. Such deployments may deploy mechanisms to map and rewrite the Route-target values on domain boundaries, using per ASBR import policies. This is no different than any other BGP VPN family. Mechanisms employed in inter-AS VPN deployments may be used with the Classful Transport family also.

The resolution schemes SHOULD allow association with multiple mapping communities. This helps with renumbering, network mergers, or transitions.

Though RD can also be rewritten on domain boundaries, deploying unique RDs is strongly RECOMMENDED, because it helps in trouble shooting by uniquely identifying originator of a route, and avoids path-hiding.

This document defines a new format of Route-Target extended-community to carry Transport Class, this avoids collision with regular Route Target namespace used by service routes.

11. Scaling considerations

11.1. Avoiding unintended spread of CT routes across domains.

RFC8212 [RFC8212] suggests BGP speakers require explicit configuration of both BGP Import and Export Policies for any EBGP sessions, in order to receive or send routes on EBGP sessions.

It is recommended to follow this for BGP CT routes. It will prohibit unintended advertisement of transport routes through out the BGP CT transport domain which may span multiple AS. This will conserve
usage of MPLS label and nexthop resources in the network. An ASBR of a domain can be provisioned to allow routes with only the Transport targets that are required by SNs in the domain.

11.2. Constrained distribution of PNHs to SNs (On Demand Nexthop)

This section describes how the number of Protocol Nexthops advertised to a SN or BN can be constrained using BGP Classful Transport and VPN RTC [RFC4684].

An egress SN MAY advertise BGP CT route for RD:eSN with two Route Targets: transport-target:0:<TC> and a RT carrying <eSN>:<TC>. Where TC is the Transport Class identifier, and eSN is the IP-address used by SN as BGP nexthop in its service route advertisements.

transport-target:0:<TC> is the new type of route target (Transport Class RT) defined in this document. It is carried in BGP extended community attribute (BGP attribute code 16).

The RT carrying <eSN>:<TC> MAY be an IP-address specific regular RT (BGP attribute code 16), IPv6-address specific RT (BGP attribute code 25), or a Wide-communities based RT (BGP attribute code 34) as described in RTC-Ext [RTC-Ext].

An ingress SN MAY import BGP CT routes with Route Target carrying: <eSN>:<TC>. The ingress SN MAY learn the eSN values either by configuration, or it MAY discover them from the BGP nexthop field in the BGP VPN service routes received from eSN. A BGP ingress SN receiving a BGP service route with nexthop of eSN SHOULD generate a RTC/Extended-RTC route for Route Target prefix <Origin ASN>:<eSN>/[80|176] in order to learn BGP CT transport routes to reach eSN. This allows constrained distribution of the transport routes to the PNHs actually required by iSN.

When path of route propogation of BGP CT routes is same as the RTC routes, a BN would learn the RTC routes advertised by ingress SNs and propagate further. This will allow constraining distribution of BGP CT routes for a PNH to only the necessary BNs in the network, closer to the egress SN.

This mechanism provides "On Demand Nexthop" of BGP CT routes, which help with scaling of MPLS forwarding state at SN and BN.
But the amount of state carried in RTC family may become proportional to number of PNHs in the network. To strike a balance, the RTC route advertisements for <Origin ASN>:<eSN>/[80|176] MAY be confined to the BNs in home region of ingress-SN, or the BNs of a super core.

Such a BN in the core of the network SHOULD import BGP CT routes with Transport Class Route Target: 0:<TC>, and generate a RTC route for <Origin ASN>:0:<TC>/96, while not propagating the more specific RTC requests for specific PNHs. This will let the BN learn transport routes to all eSN nodes. But confine their propagation to ingress-SNs.

11.3. Limiting scope of visibility of PE loopback as PNHs

It may be even more desirable to limit the number of PNHs that are globally visible in the network. This is possible using mechanism described in MPLS Namespaces [MPLS-NAMESPACES]

Such that advertisement of PE loopback addresses as next-hop in BGP service routes is confined to the region they belong to. An anycast IP-address called "Context Protocol NextHop Address" abstracts the PEs in a region from other regions in the network, swapping the PE scoped service label with a CPNH scoped private namespace label.

This provides much greater advantage in terms of scaling and convergence. Changes to implement this feature are required only on the region’s BNs and RR.

12. OAM considerations

Standard MPLS OAM procedures specified in [RFC8029] also apply to BGP Classful Transport.

The 'Target FEC Stack' sub-TLV for IPv4 Classful Transport has a Sub-Type of [TBD], and a length of 13. The Value field consists of the RD advertised with the Classful Transport prefix, the IPv4 prefix (with trailing 0 bits to make 32 bits in all), and a prefix length, encoded as follows:
The 'Target FEC Stack' sub-TLV for IPv6 Classful Transport has a Sub-Type of [TBD], and a length of 25. The Value field consists of the RD advertised with the Classful Transport prefix, the IPv6 prefix (with trailing 0 bits to make 128 bits in all), and a prefix length, encoded as follows:

```
| 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Route Distinguisher                  |
|                          (8 octets)                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         IPv6 prefix                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Prefix Length |                 Must Be Zero                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: Classful Transport IPv6 FEC

13. Applicability to Network Slicing

In Network Slicing, the Transport Slice Controller (TSC) sets up the Topology (e.g. RSVP, SR-TE tunnels with desired characteristics) and resources (e.g. polices/shapers) in a transport network to create a Transport slice. The Transport class construct described in this document represents the "Topology Slice" portion of this equation.

The TSC can use the Transport Class Identifier (Color value) to provision a transport tunnel in a specific Topology Slice.

Further, Network slice controller can use the Mapping community on the service route to map traffic to the desired Transport slice.
14. SRv6 support

This section describes how BGP CT may be used to set up inter domain tunnels of a certain Transport Class, when using Segment Routing over IPv6 (SRv6) data plane on the inter AS links or as intra-AS tunneling mechanism.

RFC8986, [SRV6-INTER-DOMAIN] specify the SRv6 Endpoint behaviors (End USD, End.BM, End.B6.Encaps and End.Replace, End.ReplaceB6, respectively). These are leveraged for BGP CT with SRv6 data plane.

The BGP Classful Transport route update for SRv6 MUST include the BGP Prefix-SID attribute along with SRv6 SID information as specified in [SRV6-SERVICES]. It may also include SRv6 SID structure for Transposition as specified in [SRV6-SERVICES]. It should be noted that prefixes carried in BGP CT family are transport layer end-points, e.g. PE loopback addresses. Thus the SRv6 SID carried in a BGP CT route is also a transport layer identifier.

This document extends the usage of "SRv6 label route tunnel" TLV to AFI=1/2 SAFI 76. "SRv6 label route tunnel" is the TLV of the BGP Prefix-SID Attribute as specified in [SRV6-MPLS-AGRWL].

15. Illustration of procedures with example topology

15.1. Topology

```
           [RR26]      [RR27]      [RR16]
           |            |            |
           +----------+----------+----------+
           |          |          |          |
           |          |          |          |
           |          |          |          |
[CE41]---[PE25]---[P28]       [P29]       [P15]       [CE31]
           |          |          |          |          |
           |          |          |          |
           |          |          |          |
           |          |          |          |
           +----------+----------+----------+

CE | region-1 | region-2 | CE
AS4       ...AS2...       AS1       AS3
           +----------+----------+----------+

41.41.41.41  ----------- Traffic Direction ----------- 31.31.31.31
```

This example shows a provider network that comprises of two Autonomous systems, AS1, AS2. They are serving customers AS3, AS4 respectively. Traffic direction being described is CE41 to CE31. CE31 may request a specific SLA, e.g. Gold for this traffic, when traversing these provider networks.

AS2 is further divided into two regions. So there are three tunnel domains in provider space. AS1 uses ISIS Flex-Algo intra-domain tunnels, whereas AS2 uses RSVP intra-domain tunnels.

The network has two Transport classes: Gold with transport class id 100, Bronze with transport class id 200. These transport classes are provisioned at the PEs and the Border nodes (ABRs, ASBRs) in the network.

Following tunnels exist for Gold transport class.

- PE25_to_ABR23_gold - RSVP tunnel
- PE25_to_ABR24_gold - RSVP tunnel
- ABR23_to_ASRBR22_gold - RSVP tunnel
- ASBR13_to_PE11_gold - ISIS FlexAlgo tunnel
- ASBR14_to_PE11_gold - ISIS FlexAlgo tunnel

Following tunnels exist for Bronze transport class.

- PE25_to_ABR23_bronze - RSVP tunnel
- ABR23_to_ASRBR21_bronze - RSVP tunnel
- ABR23_to_ASRBR22_bronze - RSVP tunnel
- ABR24_to_ASRBR21_bronze - RSVP tunnel
- ASBR13_to_PE12_bronze - ISIS FlexAlgo tunnel
- ASBR14_to_PE11_bronze - ISIS FlexAlgo tunnel

These tunnels are either provisioned or auto-discovered to belong to transport class 100 or 200.
15.2. Service Layer route exchange

Service nodes PE11, PE12 negotiate service families (SAFI 1, 128) on the BGP session with RR16. Service helpers RR16, RR26 have multihop EBGP session to exchange service routes between the two AS. Similarly PE25 negotiates service families with RR26.

Forwarding happens using service routes at service nodes PE25, PE11, PE12 only. Routes received from CEs are not present in any other nodes’ FIB in the network.

CE31 advertises a route for example prefix 31.31.31.31 with nexthop self to PE11, PE12. CE31 can attach a mapping community Color:0:100 on this route, to indicate its request for Gold SLA. Or, PE11 can attach the same using locally configured policies. Let us assume CE31 is getting VPN service from PE25.

The 31.31.31.31 route is readvertised in SAFI 128 by PE11 with nexthop self (1.1.1.1) and label V-L1, to RR16 with the mapping community Color:0:100 attached. This SAFI 128 route reaches PE25 via RR16, RR26 with the nexthop unchanged, as PE11 and label V-L1. Now PE25 can resolve the PNH 1.1.1.1 using transport routes received in BGP CT or BGP LU.

The IP FIB at PE25 will have a route for 31.31.31.31 with a nexthop thus found, that points to a Gold tunnel in ingress domain.

15.3. Transport Layer route propagation

ASBR13 negotiates BGP CT family with transport ASBRs ASBR21, ASBR22. They negotiate BGP CT family with RR27 in region 2. ABR23, ABR24 negotiate BGP CT family with RR27 in region 2 and RR26 in region 1. PE25 receives BGP CT routes from RR26. BGP LU family is also negotiated on these sessions alongside BGP CT family. BGP LU carries "best effort" transport class routes, BGP CT carries gold, bronze transport class routes.

ASBR13 is provisioned with transport class 100, RD value 1.1.1.3:10 and a transport route target 0:100. And a Transport class 200 with RD value 1.1.1.3:20, and transport route target 0:200.

Similarly, these transport classes are also configured on ASBRs, ABRs and PEs, with same transport route target, but unique RDs.
Ingress route for ASBR13_to_PE11_gold is advertised by ASBR13 in BGP CT family to ASBRs ASBR21, ASBR22. This route is sent with a NLRI containing RD prefix 1.1.1.3:10:1.1.1.1, Label B-L1 and a route target extended community transport-target:0:100. MPLS swap route is installed at ASBR13 for B-L1 with a nexthop pointing to ASBR13_to_PE11_gold tunnel.

Ingress route for ASBR13_to_PE11_bronze is advertised by ASBR13 in BGP CT family to ASBRs ASBR21, ASBR22. This route is sent with a NLRI containing RD prefix 1.1.1.3:20:1.1.1.1, Label B-L2 and a route target extended community transport-target:0:200. MPLS swap route is installed at ASBR13 for label B-L2 with a nexthop pointing to ASBR13_to_PE11_bronze tunnel.

ASBR21 receives BGP CT route 1.1.1.3:10:1.1.1.1 over the single hop EBGP session, and advertises with nexthop self (loopback address 2.2.2.1) to RR27, advertising a new label B-L3. MPLS swap route is installed for label B-L3 at ASBR21 to swap to received label B-L1 and forwards to ASBR13. RR27 advertises this BGP CT route to ABR23, ABR24.

ASBR22 receives BGP CT route 1.1.1.3:10:1.1.1.1 over the single hop EBGP session, and advertises with nexthop self (loopback address 2.2.2.2) to RR27, advertising a new label B-L4. MPLS swap route is installed for label B-L4 at ASBR22 to swap to received label B-L2 and forwards to ASBR13. RR27 advertises this BGP CT route to ABR23, ABR24.

Addpath is enabled for BGP CT family on the sessions between RR27 and ASBRs, ABRs. Such that routes for 1.1.1.3:10:1.1.1.1 with the nexthops ASBR21 and ASBR22 are reflected to ABR23, ABR24 without any path hiding. Thus giving ABR23 visibility of both available nexthops for Gold SLA.

ABR23 receives the route with nexthop 2.2.2.1, label B-L3 from RR27. The route target "transport-target:0:100" on this route acts as mapping community, and instructs ABR23 to strictly resolve the nexthop using transport class 100 routes only. ABR23 is unable to find a route for 2.2.2.1 with transport class 100. Thus it considers this route unusable and does not propagate it further. This prunes ASBR21 from Gold SLA tunnelled path.

ABR23 also receives the route with nexthop 2.2.2.2, label B-L4 from RR27. The route target "transport-target:0:100" on this route acts as mapping community, and instructs ABR23 to strictly resolve the nexthop using transport class 100 routes only. ABR23 successfully resolves the nexthop to point to ABR23_to_ASPR22_gold tunnel. ABR23 advertises this route with nexthop self (loopback address 2.2.2.3).
and a new label B-L5 to RR26. Swap route for B-L5 is installed by ABR23 to swap to label B-L4, and forward into ABR23_to_ASRBR22_gold tunnel.

RR26 reflects the route from ABR23 to PE25. PE25 receives the BGP CT route for prefix 1.1.1.3:10:1.1.1.1 with label B-L5, nexthop 2.2.2.3 and transport-target:0:100 from RR26. And it similarly resolves the nexthop 2.2.2.3 over transport class 100, pushing labels associated with PE25_to_ABR23_gold tunnel.

In this manner, the Gold transport LSP "ASBR13_to_PE11_gold" in egress-domain is extended by BGP CT until the ingress-node PE25 in ingress domain, to create an end-to-end Gold SLA path. MPLS swap routes are installed at ASBR13, ASBR22 and ABR23, when propagating the PE11 BGP CT Gold transport class route 1.1.1.3:10:1.1.1.1 with nexthop self towards PE25.

The BGP CT LSP thus formed, originates in PE25, and terminates in ASBR13, traversing over the Gold underlay LSPs in each domain. ASBR13 uses UHP to stitch the BGP CT LSP into the "ASBR13_to_PE11_gold" LSP to traverse the last domain, thus satisfying Gold SLA end-to-end.

When PE25 receives service route with nexthop 1.1.1.1 and mapping community Color:0:100, it resolves over this BGP CT route 1.1.1.3:10:1.1.1.1. Thus pushing label B-L5, and pushing as top label the labels associated with PE25_to_ABR23_gold tunnel.

15.4. Data plane view

15.4.1. Steady state

This section describes how the data plane looks like in steady state.

CE41 transmits an IP packet with destination as 31.31.31.31. On receiving this packet PE25 performs a lookup in the IP FIB associated with the CE41 interface. This lookup yeids the service route that pushes the VPN service label V-L1, BGP CT label B-L5, and labels for PE25_to_ABR23_gold tunnel. Thus PE25 encapsulates the IP packet in MPLS packet with label V-L1(innermost), B-L5, and top label as PE25_to_ABR23_gold tunnel. This MPLS packet is thus transmitted to ABR23 using Gold SLA.
ABR23 decapsulates the packet received on PE25_to_ABR23_gold tunnel as required, and finds the MPLS packet with label B-L5. It performs lookup for label B-L5 in the global MPLS FIB. This yields the route that swaps label B-L5 with label B-L4, and pushes top label provided by ABR23_to_ASBR22_gold tunnel. Thus ABR23 transmits the MPLS packet with label B-L4 to ASBR22, on a tunnel that satisfies Gold SLA.

ASBR22 similarly performs a lookup for label B-L4 in global MPLS FIB, finds the route that swaps label B-L4 with label B-L2, and forwards to ASBR13 over the directly connected MPLS enabled interface. This interface is a common resource not dedicated to any specific transport class, in this example.

ASBR13 receives the MPLS packet with label B-L2, and performs a lookup in MPLS FIB, finds the route that pops label B-L2, and pushes labels associated with ASBR13_to_PE11_gold tunnel. This transmits the MPLS packet with VPN label V-L1 to PE11, using a tunnel that preserves Gold SLA in AS 1.

PE11 receives the MPLS packet with V-L1, and performs VPN forwarding. Thus transmitting the original IP payload from CE41 to CE31. The payload has traversed path satisfying Gold SLA end-to-end.

15.4.2. Absorbing failure of primary path

This section describes how the data plane reacts when gold path experiences a failure.

Let us assume tunnel ABR23_to_ASBR22_gold goes down, such that now end-to-end Gold path does not exist in the network. This makes the BGP CT route for RD prefix 1.1.1.1:10:1.1.1.1 unusable at ABR23. This makes ABR23 send a BGP withdrawal for 1.1.1.1:10:1.1.1.1 to RR26, which then withdraws the prefix from PE25.

Withdrawal for 1.1.1.1:10:1.1.1.1 allows PE25 to react to the loss of gold path to 1.1.1.1. Let us assume PE25 is provisioned to use best-effort transport class as the backup path. This withdrawal of BGP CT route allows PE25 to adjust the nexthop of the VPN Service-route to push the labels provided by the BGP LU route. That repairs the traffic to go via best effort path. PE25 can also be provisioned to use Bronze transport class as the backup path. The repair will happen in similar manner in that case as-well.

Traffic repair to absorb the failure happens at ingress node PE25, in a service prefix scale independent manner. This is called PIC (Prefix scale Independent Convergence). The repair time will be proportional to time taken for withdrawing the BGP CT route.
16. IANA Considerations

This document makes following requests of IANA.

16.1. New BGP SAFI

New BGP SAFI code for "Classful Transport". Value 76.

This will be used to create new AFI,SAFI pairs for IPv4, IPv6 Classful Transport families. viz:

* "Inet, Classful Transport". AFI/SAFI = "1/76" for carrying IPv4 Classful Transport prefixes.

* "Inet6, Classful Transport". AFI/SAFI = "2/76" for carrying IPv6 Classful Transport prefixes.

16.2. New Format for BGP Extended Community

Please assign a new Format (Type high = 0xa) of extended community EXT-COMM [RFC4360] called "Transport Class" from the following registries:

the "BGP Transitive Extended Community Types" registry, and

the "BGP Non-Transitive Extended Community Types" registry.

Please assign the same low-order six bits for both allocations.

This document uses this new Format with subtype 0x2 (route target), as a transitive extended community.

The Route Target thus formed is called "Transport Class" route target extended community.

Taking reference of RFC7153 [RFC7153], following requests are made:

16.2.1. Existing registries to be modified

16.2.1.1. Registries for the "Type" Field

16.2.1.1.1. Transitive Types

This registry contains values of the high-order octet (the "Type" field) of a Transitive Extended Community.
Registry Name: BGP Transitive Extended Community Types

<table>
<thead>
<tr>
<th>TYPE VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 0x0a</td>
<td>Transitive Transport Class Extended</td>
</tr>
<tr>
<td>+ 0x4a</td>
<td>Non-Transitive Transport Class Extended</td>
</tr>
</tbody>
</table>

16.2.1.1.2. Non-Transitive Types

This registry contains values of the high-order octet (the "Type" field) of a Non-transitive Extended Community.

Registry Name: BGP Non-Transitive Extended Community Types

<table>
<thead>
<tr>
<th>TYPE VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 0x0a</td>
<td>Non-Transitive Transport Class Extended</td>
</tr>
<tr>
<td>+ 0x4a</td>
<td>Non-Transitive Transport Class Extended</td>
</tr>
</tbody>
</table>

16.2.2. New registries to be created

16.2.2.1. Transitive "Transport Class" Extended Community Sub-Types Registry

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 0x07.

Registry Name: Transitive Transport Class Extended Community Sub-Types

<table>
<thead>
<tr>
<th>RANGE</th>
<th>REGISTRATION PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0xBF</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>0xC0-0xFF</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUB-TYPE VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>Route Target</td>
</tr>
</tbody>
</table>

16.2.2.2. Non-Transitive "Transport Class" Extended Community Sub-Types Registry
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 0x47.

Registry Name: Non-Transitive Transport Class Extended Community Sub-Types

<table>
<thead>
<tr>
<th>RANGE</th>
<th>REGISTRATION PROCEDURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00-0xBF</td>
<td>First Come First Served</td>
</tr>
<tr>
<td>0xC0-0xFF</td>
<td>IETF Review</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUB-TYPE VALUE</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>Route Target</td>
</tr>
</tbody>
</table>

16.3. MPLS OAM code points

The following two code points are sought for Target FEC Stack sub-TLVs:

* IPv4 BGP Classful Transport
* IPv6 BGP Classful Transport

17. Security Considerations

Mechanisms described in this document carry Transport routes in a new BGP address family. That minimizes possibility of these routes leaking outside the expected domain or mixing with service routes.

When redistributing between SAFI 4 and SAFI 76 Classful Transport routes, there is a possibility of SAFI 4 routes mixing with SAFI 1 service routes. To avoid such scenarios, it is RECOMMENDED that implementations support keeping SAFI 4 routes in a separate transport RIB, distinct from service RIB that contain SAFI 1 service routes.

18. Contributors

Rajesh M
Juniper Networks, Inc.
Electra, Exora Business Park~Marathahalli - Sarjapur Outer Ring Road, Bangalore 560103
KA
India
Email: mrajesh@juniper.net
19. Acknowledgements

The authors thank Jeff Haas, John Scudder, Navaneetha Krishnan, Ravi M R, Chandrasekar Ramachandran, Shraddha Hegde, Richard Roberts, Krzysztof Szarkowicz, John E Drake, Srijani Sangli, Vijay Kestur, Santosh Kolenchery, Robert Raszuk, Ahmed Darwish for the valuable discussions and review comments.

The decision to not reuse SAFI 128 and create a new address-family to carry these transport-routes was based on suggestion made by Richard Roberts and Krzysztof Szarkowicz.

20. Normative References

[MPLS-NAMESPACES]

[PCEP-RSVP-COLOR]


Authors’ Addresses

Kaliraj Vairavakkalai (editor)
Juniper Networks, Inc.
1133 Innovation Way,
Sunnyvale, CA 94089
United States of America
Email: kaliraj@juniper.net

Natrajan Venkataraman
Juniper Networks, Inc.
1133 Innovation Way,
Sunnyvale, CA 94089
United States of America
Email: natv@juniper.net
Balaji Rajagopalan
Juniper Networks, Inc.
Electra, Exora Business Park–Marathahalli – Sarjapur Outer Ring Road,
Bangalore 560103
KA
India
Email: balajir@juniper.net

Gyan Mishra
Verizon Communications Inc.
13101 Columbia Pike
Silver Spring, MD 20904
United States of America
Email: gyan.s.mishra@verizon.com

Mazen Khaddam
Cox Communications Inc.
Atlanta, GA
United States of America
Email: mazen.khaddam@cox.com

Xiaohu Xu
Capitalonline.
Beijing
China
Email: xiaohu.xu@capitalonline.net

Rafal Jan Szarecki
Google.
1160 N Mathilda Ave, Bldg 5,
Sunnyvale,, CA 94089
United States of America
Email: szarecki@google.com

Deepak J Gowda
Extreme Networks
55 Commerce Valley Drive West, Suite 300,
Thornhill, Toronto, Ontario L3T 7V9
Canada
Email: dgowda@extremenetworks.com
Chaitanya Yadlapalli
ATT
200 S Laurel Ave,
Middletown,, NJ 07748
United States of America
Email: cy098d@att.com
Abstract

[I-D.ietf-idr-link-bandwidth] defines a BGP link bandwidth extended community attribute, which can enable devices to implement unequal-cost load-balancing. However, the bandwidth value encapsulated by the extended community attribute is of the floating-point type, which is inconvenient to use. In this document, a set of new types of link bandwidth extended community are introduced to facilitate the configuration and calculation of link bandwidth.

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 6 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ............................................. 2
2. Link Bandwidth Extended Community .................... 3
3. Deployment Considerations ............................... 3
4. IANA Considerations ..................................... 4
5. Security Considerations .................................. 4
6. Acknowledgements ........................................ 4
7. References ................................................ 4
   7.1. Normative References ................................. 4
   7.2. References ........................................... 4
Authors’ Addresses ......................................... 4

1. Introduction

In [I-D.ietf-idr-link-bandwidth], the link bandwidth extended community attribute is added to implement unequal-cost load balancing based on the bandwidth on a path. As defined in the draft, the bandwidth of a link is expressed in 4-octets in IEEE floating-point format.

In practice, the use of this floating-point format may result errors in configuration and computation. When an operator needs to manually specify the bandwidth, you also need to consider the conversion from the bandwidth value to the floating-point number. This mode is not user-friendly, especially when the routing policy is used for bandwidth matching.

This document introduce a more intuitive expression of link bandwidth in BGP. It uses an unsigned long integer value to describe the link bandwidth value. This is easier for operators to use and understand, and can avoid configuration and computation errors.
2. Link Bandwidth Extended Community

The type of Link Bandwidth Extended Community is 0x40, and the subtype is 0x04. In the attribute value, the global administrator subfield is set to the AS number of the route to which the Link Bandwidth attribute is added. In the local administrator subfield, the link bandwidth value [I-D.ietf-idr-link-bandwidth] is set to the IEEE floating-point type.

A new type of IPv6 Address Specific Extended Community[RFC5701] is added in this document. The ASN field of this attribute is set to the AS number of the route to which the link bandwidth attribute is added. The Link Bandwidth value field (8 bytes) is set to the link bandwidth. The following extended contents are added:

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| 0x03 or 0x43 |   Sub-Type    |            Reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     8 bytes Link Bandwidth value (cont.)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     8 bytes Link Bandwidth value (cont.)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         4 bytes ASN                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Reserved                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

* The value of the high-order octet of the extended Type, refer to [RFC4360], It is recommended that 0x03 and 0x43 be used.

* New Link Bandwidth, subtype is TBD. The value of the Link Bandwidth subfield is an unsigned long integer, in bytes per second.

The subtypes defined here can be used for both optional transitive and non-transitive extended community attributes.

3. Deployment Considerations

The extended link bandwidth extended community attribute in this document should not be used together with the standard link bandwidth extended community attribute. If a route carries both the standard link bandwidth extended community attribute and the unit link bandwidth extended community attribute, the standard link bandwidth extended community attribute is ignored.
In actual deployment, if a routing policy is used to match link bandwidth attributes, you can directly perform exact value matching.

4. IANA Considerations

This document defines a specific application of the two-octet AS specific extended community. IANA is requested to assign new sub-types for both non-transitive and transitive extended communities.

<table>
<thead>
<tr>
<th>SubType</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Link Bandwidth EC in bytes per second</td>
</tr>
</tbody>
</table>

5. Security Considerations

There are no additional security risks introduced by this design.

6. Acknowledgements

7. References

7.1. Normative References


7.2. References


Authors’ Addresses
Dissemination of BGP Flow Specification Rules for APN  
draft-peng-apn-bgp-flowspec-01

Abstract

A BGP Flow Specification is an n-tuple consisting of several matching criteria that can be applied to IP traffic. Application-aware Networking (APN) is a framework, where APN data packets convey APN attribute including APN ID and/or APN Parameters. The dynamic Flow Spec mechanism for APN is designed for the new applications of traffic filtering in an APN domain as well as the traffic control and actions at the policy enforcement points in this domain. These applications require coordination among the ASes within a service provider.

This document specifies a new BGP Flow Spec Component Type in order to support APN traffic filtering. The match field is the APN ID. It also specifies traffic filtering actions to enable the creation of the APN ID in the outer tunnel encapsulation when matched to the corresponding Flow Spec rules.

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 31 October 2022.
Copyright Notice

Copyright (c) 2022 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
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 Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ............................................. 3
2. Requirements Language ................................. 4
3. Terminologies ........................................... 4
4. Flow Specifications for APN ............................. 4
5. Component Type for APN ................................ 5
   5.1. APN ID - Type TBD1 ............................... 5
   5.2. Encoding Example ................................. 5
6. Traffic Filtering ......................................... 6
   6.1. Ordering of Flow Specifications ................. 6
   6.2. Encoding format of the Grouping Identifier Extend Community Sub-Type TBD2 ............... 7
   6.3. Usage Principles ................................ 7
   6.4. Usage example ................................... 8
7. Traffic Filtering Actions ............................... 10
   7.1. Traffic Marking (traffic-marking-apn) Sub-Type TBD3 .... 11
   7.2. Traffic Marking (traffic-marking-apn-partial) Sub-Type TBD4 ................................. 12
   7.3. Inherit (inherit-apn) Sub-Type TBD5 ............. 13
   7.4. Stitch (stitch-apn) Sub-Type TBD6 ............... 13
8. IANA Considerations ..................................... 14
   8.1. Flow Spec Component - APN ID .................... 14
   8.2. Opaque Extended Community - Grouping Identifier .......... 15
   8.3. Extended Community Flow Specification Actions ........ 15
9. Acknowledgements ......................................... 15
10. Security Considerations ................................. 16
11. References .............................................. 16
   11.1. Normative References ............................ 16
   11.2. Informative References ......................... 17
Authors’ Addresses .......................................... 18
1. Introduction

A Flow Specification (Flow Spec) is an n-tuple consisting of several matching criteria that can be applied to IP traffic [RFC8955]. The Flow Spec conveys match conditions (each may include several components) which are encoded using MP_REACH_NLRI and MP_UNREACH_NLRI attributes [RFC4760], while the associated actions such as redirect and traffic marking are encoded in BGP Extended Communities [RFC4360][RFC5701]. The IPv4 NLRI component types and traffic filtering actions sub-types are described in [RFC8955], while the IPv6 related are described in [RFC8956].

[I-D.ietf-idr-flowspec-l2vpn] extends the flow-spec rules and actions for Ethernet Layer 2 and L2VPN. The corresponding (AFI, SAFI) pairs are defined by IANA, respectively. [I-D.hares-idr-flowspec-v2] specifies BGP Flow Specification Version 2.

Application-aware Networking (APN) is introduced in [I-D.li-apn-framework] and [I-D.li-apn-problem-statement-usecases]. APN data packets convey the APN attribute (incl. APN ID and/or APN Parameters). The APN ID is a structured value, treated as an opaque object in the network, to which the network operator applies policies in various nodes/service functions along the path so to provide corresponding services. For an IPv6 network, a design proposal of such structured value is provided by [I-D.li-apn-header][I-D.li-apn-ipv6-encap]. The APN attribute can be encapsulated in various data planes adopted within a Network Operator controlled limited domain, e.g. IPv6, MPLS, and other tunnel technologies, which wait to be further specified.

With APN, it becomes possible to apply various policies in different nodes along a network path onto a traffic flow overall in a more efficient way, that is, at the headend to steer into corresponding path, at the midpoint to collect corresponding performance measurement data, and at the service function to execute particular policies. Prior to APN, there was no efficient way to realize this composite network service provisioning along the path.

This document specifies a new BGP Flow Spec Component Type to support APN traffic filtering. The match field is the APN APN ID [I-D.li-apn-framework]. It also specifies traffic filtering actions to enable the creation of the APN ID in the outer tunnel encapsulation when matched to the corresponding Flow Spec rules.

Depends upon specific deployment requirements, the functions specified in this draft can also be applied on BGP Flow Specification Version 2, which will be specified in the future versions.
2. 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 RFC 2119 [RFC2119] RFC 8174 [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Terminologies

APN: Application-aware Networking

APN ID: APN Identifier

AS: Autonomous System

Flow Spec: Flow Specification


4. Flow Specifications for APN

The APN framework is introduced in [I-D.li-apn-framework]. The Flow Spec for APN is shown in Figure 1, that is, the Controller is used to set up BGP connection with the policy enforcement points in an APN domain.

```
+---------------+      \
/      +---------------+      \
|App x|-
\-|APN |-/    |App y|--/--- Network Operator Controlled ---|
+-----+      |     +-----+      |

Figure 1. Flow Spec for APN
```
5. Component Type for APN

The IPv4 NLRI component types are defined in [RFC8955], while the IPv6 related are specified in [RFC8956]. This document defines a new component type for APN.

5.1. APN ID - Type TBD1

Encoding: <type (1 octet), length (1 octet), mask (variable), APN ID (variable)>

Defines the APN ID to match. The mask is used to indicate the bits of the APN ID carried in the packet which are used to match against the APN ID value in this Flow Spec component.

type (1 octet): This indicates the new component type TBD1.

length (1 octet): This indicates the length of the mask and the length of the APN ID. The mask and the APN ID have the same length.

mask (variable): This indicates the bits of the APN ID carried in the data packet which are used to match.

APN ID (variable): This indicates the APN ID that is used for the match.

5.2. Encoding Example

Since the APN ID is a structured value, the mask in the Flow Spec is used to enable flexible matching of the particular parts of the APN ID.

As an example, shown in Figure 2, the APN ID in the data packet contains two parts, the APP Group ID (0x300A) and User Group ID (0x0C08). In the Flow Spec, the mask is 0xFFFF0000 and the APN ID is 0x300A0000. Processing the match of the APN ID component is done by using the mask (0xFFFF0000) to indicate the bits of the APN ID carried in the packet to be matched against the one carried in the Flow Spec (0x300A0000). The result of this example is a successful match.
6. Traffic Filtering

Traffic filtering policies have been traditionally considered to be relatively static. The dynamic Flow Spec mechanism for APN is designed for the new applications of traffic filtering in an APN domain as well as the traffic control and actions on the policy enforcement points in this domain. These applications require coordination among the ASes within a service provider. The new component and encoding are defined in Section 4. The actions are defined in this section.

6.1. Ordering of Flow Specifications

More than one Flow Specification rule may match a particular traffic flow at a node. The co-existing rules are mixed and need to be effectively organized. However, there is still no efficient way to achieve such classification. Thus, it is necessary to specify the grouping mechanism for the Flow Specification rules to be matched in a desired order as well as the actions being applied to a particular traffic flow. This ordering function is such that it does not depend on the arrival order of the Flow Specification via BGP and thus is consistent in the network [RFC8955].

The definition of this ordering is very important to the Flow Spec for APN because of the following reasons.

1. There can be other co-existing Flow Spec rules (e.g. based on 5-tuple) rather than only APN Flow Spec rules (i.e. based on APN ID).

2. The different parts of the APN ID can be determined by the different Flow Spec rules.
Therefore, the ordering of the Flow Spec rules for APN needs to be clearly specified.

6.2. Encoding format of the Grouping Identifier Extend Community Sub-Type TBD2

We define a Grouping Identifier Opaque Extend Community [RFC4360] (Sub-Type = TBD2) carrying both Group ID (2 octets) and Sub-group ID (2 octets) and indicating the grouping of the Flow Spec rules it accompanies.

The encoding format of the Grouping Identifier Opaque Extend Community is as follows.

```
0                   1
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| 0x03 or 0x43  |Sub-Type = TBD2|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|           Group ID            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|         Sub-Group ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Figure 3: Encoding of the Grouping Identifier Extend Community

6.3. Usage Principles

The following principles are defined.

1. Within a sub-group, the order is the same as the previously defined.

* If the traffic-action Extended Community is carried and the Terminal Action (T, bit 47) [RFC8955] is not set, when one condition in this sub-group is matched, the evaluation of any subsequent flow specifications within this sub-group stops; if T is set, then the evaluation continues;

* If the traffic-action Extended Community is not carried, when one condition in this sub-group is matched, the evaluation of any subsequent flow specifications within this sub-group stops;
2. Between sub-groups, the sub-group is ordered by increasing Sub-
group ID, when the evaluation in one sub-group stops or finishes, it
will start the evaluation in the following sub-group if there are any
sub-groups.

3. Between groups, the group is ordered by increasing Group ID, if
at least one condition in this group is matched, when the evaluation
of the flow specifications within the group reaches the end, the
evaluation stops so the evaluation of the following group(s) will not
start.

6.4. Usage example

At the APN Edge where the APN ID is created based on the Flow
Specifications and encapsulated in the outer tunnel header
[I-D.li-apn-framework], more than one Flow Specification rule
condition may match a particular traffic flow. The different parts
of the APN ID can be determined by the different Flow Spec rules.
For example, as shown in Figure 4, the App Group ID is created by
matching the 5-tuple components (e.g. destination IP address and
transport layer ports), the User Group ID is created by matching the
access ports, and the Reserved (R.) Group ID is created by matching
the 5-tuple components.

Moreover, there are also other co-existing Flow Spec rules mixed at
the node rather than only APN Flow Spec rules (i.e. based on APN ID).
All the rules need to be effectively organized and applied to the
particular traffic flow in a desired order.

In Figure 4, the Flow Specification rules for APN and other existing
rules are categorized into two groups, and given Group ID = 1 and 2,
respectively. The Flow Specification rules for creating different
parts of the APN ID are categorized into three sub-groups, and given
Sub-Group ID = 1, 2, and 3, respectively.

Based on the usage principles described in the above section, for the
case of APN as shown in Figure 4, the usage principles are as
follows,

1. Within a sub-group, the order is the same as the previously
defined.
* If the traffic-action Extended Community is carried and the Terminal Action (T, bit 47) [RFC8955] is not set, when one condition in this sub-group is matched, the evaluation of any subsequent flow specifications within this sub-group stops and the App Group ID is created; if T is set, then the evaluation continues and the App Group ID will be created if there is a match within this sub-group;

* If the traffic-action Extended Community is not carried, when one condition in this sub-group is matched, the evaluation of any subsequent flow specifications within this sub-group stops and the App Group ID is created;

2. Between sub-groups, the sub-group is ordered with Sub-group ID, when the evaluation in the Sub-group ID = 1 stops or finishes, it will start the evaluation in the following Sub-group ID = 2 and create the User Group ID if matched, and then the Sub-group ID = 3 to create the R. Group ID if matched.

3. Between groups, the group is ordered with Group ID, if at least one condition in this Group ID = 1 is matched, when the evaluation of the flow specifications within the group reaches the end, the evaluation stops and the APN ID is created. The evaluation of the following group(s) will not start, that is, the Group ID = 0 will not be evaluated.

```
Group ID = 1, Sub-Group ID = 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule (5-tuple) |         App Group ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule (5-tuple) |         App Group ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule (5-tuple) |         App Group ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule (5-tuple) |         App Group ID          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Group ID = 1, Sub-Group ID = 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule ( ports ) |         User Group ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule ( ports ) |         User Group ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule ( ports ) |         User Group ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...                          ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Rule ( ports ) |         User Group ID         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
Figure 4: Usage of Grouping Identifier Extended Community for APN

7. Traffic Filtering Actions
### Table: APN FlowSpec Dissemination Sub-Types

<table>
<thead>
<tr>
<th>Community Sub-Type</th>
<th>action</th>
<th>encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD3</td>
<td>traffic-marking-apn</td>
<td>4/16-octet APN ID 1-octet IPv6 (ExH) Type 1-octet Reserved</td>
</tr>
<tr>
<td>TBD4</td>
<td>traffic-marking-apn-partial</td>
<td>4/16-octet Bitmask 4/16-octet APN ID 1-octet IPv6 (ExH) Type 1-octet Reserved</td>
</tr>
<tr>
<td>TBD5</td>
<td>inherit-apn</td>
<td>4/16-octet Bitmask 1-octet IPv6 (ExH) Type 1-octet Reserved</td>
</tr>
<tr>
<td>TBD6</td>
<td>stitch-apn</td>
<td>4/16-octet Bitmask 4/16-octet APN ID 1-octet IPv6 (ExH) Type 1-octet Reserved</td>
</tr>
</tbody>
</table>

#### 7.1. Traffic Marking (traffic-marking-apn) Sub-Type TBD3

The traffic-marking-apn Extended Community instructs a system to create the APN ID and encapsulate it in the indicated outer tunnel header of a transiting IP packet.

In this case, the tunnel encapsulation header is IPv6, possibly followed by an extension header (ExH). The corresponding Extended Community [RFC5701] is encoded as follows:

```
+----------------------------------+
|                                 |
| IPv6 (ExH) | reserved | 1-octet IPv6 (ExH) Type 1-octet Reserved |
| 4/16-octet Bitmask 4/16-octet APN ID |
+----------------------------------+
```

```plaintext
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| APN ID |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IPv6 (ExH) | reserved |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
APN ID: 4/16 octets, APN ID value to be created and encapsulated in the indicated outer tunnel header of the transiting IP packet.

IPv6 (ExH): 1 octet, the type of each IPv6 extension header \[RFC8200\] [RFC2780] [RFC5871] is directly reused to indicate the outer tunnel to be used to encapsulate the APN ID.

reserved: 1 octet, MUST be set to 0 on encoding and MUST be ignored during decoding.

7.2. Traffic Marking (traffic-marking-apn-partial) Sub-Type TBD4

The traffic-marking-apn-partial Extended Community instructs a system to use the bitmask indicating the bits of the APN ID to be encapsulated in the indicated outer tunnel header of a transiting IP packet. The ultimately constructed APN ID may comprise of several parts obtained by the matches of different rules, and it is encapsulated in the indicated outer tunnel header.

In this case, the tunnel encapsulation header is IPv6, possibly followed by an extension header (ExH). The corresponding Extended Community [RFC5701] is encoded as follows:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Bitmask                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              APN ID                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  IPv6 (ExH)   |   reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bitmask: 4/16 octets, the same length as the APN ID, indicating the bits of the APN ID to be encapsulated in the indicated outer tunnel header.

APN ID: 4/16 octets, the APN ID value to be created and encapsulated in the indicated outer tunnel header of the transiting IP packet.
IPv6 (ExH): 1 octet, the type of each IPv6 extension header
[RFC8200][RFC2780][RFC5871] is directly reused to indicate the outer
tunnel to be used to encapsulate the APN ID.

reserved: 1 octet, MUST be set to 0 on encoding and MUST be ignored
during decoding.

7.3. Inherit (inherit-apn) Sub-Type TBD5

The inherit-apn Extended Community instructs a system to use the
Bitmask to "and" operate on the existing APN ID of a transiting IP
packet and encapsulate the inherited APN ID in the indicated outer
tunnel header.

In this case, the tunnel encapsulation header is IPv6, possibly
followed by an extension header (ExH). The corresponding Extended
Community [RFC5701] is encoded as follows:

```
      0                   1                   2                   3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                             Bitmask                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|  IPv6 (ExH)   |   reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Bitmask: 4/16 octets, the same length as the APN ID, to "and" operate
on the existing APN ID of a transiting IP packet.

IPv6 (ExH): 1 octet, the type of each IPv6 extension header
[RFC8200][RFC2780][RFC5871] is directly reused to indicate the outer
tunnel to be used to encapsulate the APN ID.

reserved: 1 octet, MUST be set to 0 on encoding and MUST be ignored
during decoding.

7.4. Stitch (stitch-apn) Sub-Type TBD6

The stitch-apn Extended Community instructs a system to "and" the
Bitmask with the existing APN ID of a transiting IP packet to get the
part to be further encapsulated, and "and" the negation of the
Bitmask with the APN ID in the Flow Spec and get the other part to be
further encapsulated. The stitched APN ID is encapsulated in the
indicated outer tunnel header. That is to say, the Bitmask specifies
the bits of the received APN ID to be replaced by the corresponding bits from the APN ID in the action sub-TLV value to produce a new outer APN ID. The other bits of the received APN ID are copied to the new outer AP ID.

In this case, the tunnel encapsulation header is IPv6, possibly followed by an extension header (ExH). The corresponding Extended Community [RFC5701] is encoded as follows:

```
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Bitmask                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          APN ID                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IPv6 (ExH) |   reserved    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bitmask: 4/16 octets, the same length as the APN ID, used to operate on the APN ID (both carried in the transiting IP packet and in the Flow Spec).

APN ID: 4/16 octets, the APN ID value to be created and encapsulated in the indicated outer tunnel header of the transiting IP packet.

IPv6 (ExH): 1 octet, the type of each IPv6 extension header [RFC8200][RFC2780][RFC5871] is directly reused to indicate the outer tunnel to be used to encapsulate the APN ID.

reserved: 1 octet, MUST be set to 0 on encoding and MUST be ignored during decoding.

8. IANA Considerations

8.1. Flow Spec Component – APN ID

IANA is requested to assign a value in the Flow Specification Component Types Registry as follows:
8.2. Opaque Extended Community - Grouping Identifier

The Grouping Identifier Opaque Extended Community is defined in this document and it is requested that a Sub-Type = TBD2 be assigned as follows.

| Value | Name               | Reference     |
|-------+--------------------|---------------+
| TBD2  | Grouping Identifier | This document |

8.3. Extended Community Flow Specification Actions

The Extended Community Flow Specification Actions are defined in this document and it is requested that corresponding Sub-Types as shown in the following table be assigned.

| Sub-Type Value | Name                         | Reference     |
|---------------+------------------------------|---------------+
| TBD3           | traffic-marking-apn           | This document |
| TBD4           | traffic-marking-apn-partial   | This document |
| TBD5           | inherit-apn                  | This document |
| TBD6           | stitch-apn                   | This document |

9. Acknowledgements

The authors would like to thank the careful reviews and valuable comments from Haibo Wang, Shunwan Zhuang, Stefano Previdi, and Donald Eastlake.
10. Security Considerations

The security considerations are the same as [RFC8955], [RFC8956], and [I-D.li-apn-framework].

11. References

11.1. Normative References

[I-D.hares-idr-flowspec-v2]

[I-D.li-apn-framework]

[I-D.li-apn-header]

[I-D.li-apn-ipv6-encap]


11.2. Informative References

[I-D.ietf-idr-flowspec-l2vpn]

[I-D.li-apn-problem-statement-usecases]


BGP Metric Credit Based Routing
draft-peng-idr-bgp-metric-credit-00

Abstract

This document defines an optional metric related BGP path attribute that can be advertised with BGP intent routes, to provide more clear intent information used for the next-hop 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 July 1, 2022.

Copyright Notice

Copyright (c) 2021 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

In large-scale networks across multiple domains, BGP can be used to advertise intent route and provide end-to-end intent aware path. According to corresponding intent information, a BGP intent route selects the expected underlay path for the next hop resolution. The underlay path may be an MPLS-TE tunnel, an SR policy, an IGP Flex-algo route, or a direct link, established segment by segment based on the same intent.

There are many methods to make BGP carry intent information during route advertisement.

[I-D.zhou-idr-inter-domain-lcu] describe a simple method, termed as BGP-LU color, combined with [RFC7911], to create multiple BGP-LU color path to the same destination, using color as intent identifier.

[I-D.kaliraj-idr-bgp-classful-transport-planes] defines new "Classful Transport" SAFI NLRI and "Transport Class" Route Target extended community, to advertise intent based routes with related Transport Class identifier as intent identifier.

[I-D.dskc-bess-bgp-car] defines new "BGP CAR" SAFI NLRI that contains a Color field as intent identifier to advertise intent based routes.
[I-D.lp-lsr-bgp-algorithm] introduces Algorithm Extended Community to advertise BGP algorithm routes, using algorithm as intent identifier.

Once a BGP speaker received an intent route advertised from neighbor, it will check the related intent template in local. The intent template could be a Color template, a Flex-algorithm Definition, or some other modules. The intent template contains a set of constraints, such as the reserved bandwidth to be provided in the path, the boundary minimum and maximum delay, the boundary delay jitter, the boundary packet loss rates, including or excluding specific nodes or links, limiting the calculation of the path in a specific virtual network, and so on. The detailed intent information itself are not recommended to be directly carried in the advertised route.

On the ingress PE node in the network, a BGP intent route to the egress PE will be selected according to the service SLA. The intent template configured on the ingress PE node is generally consistent with the service SLA. However this does not mean that the intent template configured on the intermediate nodes should also be consistent with the service SLA. For example, the SLA of the service is to "provide a path with an upper delay boundary of 100ms from ingress PE to egress PE". In this case, the upper delay of 100ms refers to end-to-end delay constraint, not for each segment. A possible method is to include different delay constraints in the intent template configured on different BGP speakers along the path. However, this static configuration method has shortcomings, because an intent template is not necessarily bound to a specific end-to-end path and may be used for multiple paths.

It can be seen that the information contained in the intent template is not enough to represent the complete intent. This often occurs on cumulable metric attributes.

This document describes extensions to carry the metric credit information in the BGP route advertisement, as a supplement to the intent template, so that the BGP speaker receiving the BGP intent route can establish (or select) the underlay path to the downstream BGP speaker with more accurate intent indication.

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. Extensions for BGP Route Metric

This section describes some extensions related to metric, so that when BGP speaker advertise intent route to upstream neighbors, it carries optional metric type, metric, and metric credit information.

2.1. Extensions for Metric Type and Metric

[RFC7311] defines AIGP Attribute to represent basic IGP default metric type and its cumulative value.

[I-D.ssangli-idr-bgp-generic-metric-aigp] defines extensions to the AIGP attribute to carry Generic Metric sub-types, to meet more metric types, such as Min Unidirectional delay metric defined in [RFC8570], TE Default Metric defined in [RFC5305], Bandwidth Metric defined in [I-D.ietf-1sr-flex-algo-bw-con], etc.

This document reuse the above definitions.

2.2. Extensions for Metric Credit

A new path attribute type (TBD): METRIC-CREDIT attribute, is introduced in this document. It is an optional non-transitive attribute, which is used to specify end-to-end total metric credit, estimated BGP hop counts, and metric credit piece information.

The format of attribute value is shown below.

```
+----------------------------------------------------+
| Count of Sources (1 octet)                          |
+----------------------------------------------------+
| Flags (1 octet)                                    | <--- 1st Source
+----------------------------------------------------+
// Network Address of Source (variable)               //
+----------------------------------------------------+
| Total Metric Credit for Source (4 octets)           |
+----------------------------------------------------+
| Estimated BGP Hops Count for Source (1 octet)       |
+----------------------------------------------------+
// Current Hop Number (1 octet)                       //
+----------------------------------------------------+
// Metric Credit Piece [Hops Count] (variable)        //
+----------------------------------------------------+
// ... For other Sources ...                         // <--- 2nd Source
+----------------------------------------------------+
```

Figure 1: Metric Credit Path Attribute Format
Count of Sources (1 octet): Indicates how many sources (i.e. ingress PE) have corresponding metric credit information.

Flags (1 octet): Currently three flags are defined.

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

Figure 2

S-Flag: Source Address Flag. The Network Address of Source field is included when S-Flag is set, and not included when unset.

F-Flag: Address Family Flag. Indicate the address family of the Network Address of Source field. 0 for 32 bits IPv4 address, 1 for 128 bits IPv6 address.

P-Flag: Piece Flag. Indicates whether specific metric credit piece information (Current Hop Number field and Metric Credit Piece[] field) is included.

Network Address of Source (variable octets): Represents the IP address of the source. When S-flag is 0, this field does not exist; When S-flag is 1 and F-flag is 0, this field is a 32-bits IPv4 address; When S-flag is 1 and F-flag is 1, this field is a 128-bits IPv6 address. If this field does not exist, it means that the metric credit information is independent of the specific source. This generally occurs in the scenario that want to control the establishment of intent paths according to the metric credit in coarse granularity. For example, in a small administrative domain, when the egress PE advertise BGP intent routes to multiple ingress PEs, only a single unified metric credit information for all source is carried.

Total Metric Credit for Source (4 octets): Represents the total metric credit from a specific ingress PE (or any source, when the Network Address of Source field does not exist) to egress PE.

Estimated BGP Hops Count for Source (1 octet): Represents the estimated number of BGP hops from a specific ingress PE (or any source, when the Network Address of Source field does not exist) to egress PE. Only those BGP speakers who modified BGP next hop to self when receiving BGP intent route are counted.

Current Hop Number (1 octet): Represents the current index of the array Metric Credit Piece[]. Note that when P-flag is 0, the
"Current Hop Number" and "Metric Credit Piece[]" fields do not exist. In the BGP intent route advertisement originated from egress PE, the initial value of Current Hop Number is 0; When the BGP intent route advertisement received by an upstream BGP speaker and the BGP speaker modifies the BGP next hop to self, it read the item from the array Metric Credit Piece[] using Current Hop Number as index, to obtain the metric credit piece that is available from the current BGP speaker to the neighbor of the downstream BGP speaker. When the BGP speaker continue to advertise the route to the upstream BGP speaker neighbor and modify BGP next hop to self, it increase the Current Hop Number by 1. Note that when using Current Hop Number as index to read items from the array Metric Credit Piece[], it must avoid array out of bounds.

Metric Credit Piece[] (variable octets): Is an array. The number of items is specified by Estimated BGP Hops Count for Source. Each item occupies 3 bytes (in microseconds) for each BGP speaker along the intent path. There are strict restrictions on the use of Metric Credit Piece[], i.e., BGP intent route must be advertised to a specific ingress PE hop by hop in strict accordance with the propagated path expected by egress PE. When an exception occurs, e.g., a BGP speaker finds that the Current Hop Number value in the received BGP intent route is greater than or equal to the Estimated BGP Hops Count for Source value, then it must stop using Metric Credit Piece[] for BGP next hop resolution.

3. Process of Received Metric Credit

When a BGP speaker receives BGP intent route with metric credit information from neighbors, it obtains the intent identifier from the route advertisement, looks up the intent template locally to obtain the detailed intent information, and selects underlay path according to the intent information.

The metric credit attribute information contained in the route advertisement can provide a suggested metric credit value for this BGP speaker to select underlay path destined to the downstream neighbor BGP speaker.

3.1. Only Total Metric Credit Provided

If the metric credit attribute only contains the total metric credit related to the source, but does not contain the explicit metric credit piece information, then for each source:

Let \( \text{metric_residual_value} = "\text{Total Metric Credit for Source}" - \ "\text{metric of AIGP Attribute}" \)
Let average_metric_credit_value = "Total Metric Credit for Source" / "Estimated BGP Hops Count for Source"

The suggested metric credit for the current segment is the minimum value of metric_residual_value (note: negative values need to be excluded) and average_metric_credit_value for all sources.

3.2. Metric Credit Piece Provided

If the BGP intent route advertisement also contains explicit metric credit piece information, then:

Let explicit_metric_credit_piece_value = Metric Credit Piece[Current Hop Number]

The suggested metric credit for the current segment is the minimum value of metric_residual_value (note: negative values need to be excluded) and explicit_metric_credit_piece_value for all sources.

It is recommended that this option is mainly used in the case that only a single source related metric credit piece information is included in the intent route that is advertised to a single ingress PE. Multiple source related metric credit piece information may bring conflicts and cause inaccurate suggested metric credit.

To include explicit BGP speaker list in the route advertisement and specify metric credit piece for each segment, will be described in the next version.

3.3. Select Underlay Path According to Suggested Metric Credit

The purpose of suggested metric credit is to restrict that the cumulative metric of the resolved underlay path cannot exceed the expected value. However, in some cases, if the BGP speaker finds that there is no underlay path that can meet the suggested value, this constraint can be relaxed appropriately by local policy.

For the BGP intent route installed on the received BGP speaker, the metric value is updated to the metric of AIGP attribute plus the cumulative metric of the underlay path for the corresponding metric type.

4. Examples
4.1. Example of Intent with Average Metric Credit

As shown in Figure 3, it contains two IGP domains. BGP neighbors are established between PE1 and ABR, ABR and PE2, to advertise BGP intent routes. Egress PE2 advertise its loopback route, loopback-PE2, to ABR through BGP, and carries intent identifier and metric credit attribute.

```
+---------P1---------+    +---------P4---------+
 |                 |    |                 |
 PE1---------P2---------ABR--------P5--------PE2
 |                 |    |                 |
 +---------P3---------+    +---------P6---------+
```

|<----- IGP domain 1 ---->|<----- IGP domain 2 ---->|

TE-path-11: PE1-P1-ABR   TE-path-12: ABR-P4-PE2
TE-path-21: PE1-P3-ABR   TE-path-22: ABR-P6-PE2

Figure 3: Inter-area Intent Routing

There are two services to communicate between ingress PE1 and egress PE2. The intent of the first service is that the end-to-end total delay of the transport path used shall not exceed 10ms, for the intent of the second service, it is 100ms. Since two intent aware paths need to be instantiated between the same source/destination for two services, in general, two intent identifiers, intent-1000 and intent-2000, need to be configured on the egress PE2.

The intent-1000 template may have the following configuration:

- metric-type: Unidirectional Link Delay (ms)
- total-metric-credit: 10 (ms)
- metric-credit enabled

The intent-2000 template may have the following configuration:

- metric-type: Unidirectional Link Delay (ms)
- total-metric-credit: 100 (ms)
- metric-credit enabled

The above intent template are also configured in other BGP speakers (i.e., ABR and PE1). However, it should be noted that since the
intent template contains the metric credit enabled command, these BGP speakers, for the matched intent routes that are not originated from themselves, will not select the underlay path to the downstream BGP speaker neighbor only according to the total metric contained in the intent template. Instead, they should obtain the suggested metric credit from the received BGP intent route, and then select the appropriate underlay path that meets the suggested metric credit.

The total metric credit included in the intent template is mainly used for egress PE2 to generate metric credit information that is encoded in the originated intent route. Other BGP speakers (non initiator) cannot directly use it to select underlay paths.

PE2 advertises two BGP intent routes, <prefix = loopback-PE2, intent-id = 1000> and <prefix = loopback-PE2, intent-id = 2000>, which are advertised to ABR respectively. The BGP next hop in the advertised route is set to PE2.

The metric related information encoded in <prefix = loopback-PE2, intent-id = 1000> is:

metric-type: Unidirectional Link Delay
metric: assume an initial value of 0
metric-credit information:
    Count of Sources: 1
    S-Flag = 1, F-Flag = 0, P-Flag = 0
    Network Address of Source: loopback-PE1
    Total Metric Credit for Source: 10
    Estimated BGP Hops Count for Source: 2

Similarly, the metric related information encoded in <prefix = loopback-PE2, intent-id = 2000> is:

metric-type: Unidirectional Link Delay
metric: assume an initial value of 0
metric-credit information:
    Count of Sources: 1
S-Flag = 1, F-Flag = 0, P-Flag = 0
Network Address of Source: loopback-PE1
Total Metric Credit for Source: 100
Estimated BGP Hops Count for Source: 2

After receiving the first route advertisement, ABR knows that the suggested metric credit from itself to the downstream BGP speaker neighbor PE2 is 5ms, i.e., the average metric credit 5, the residual metric credit 10, taking the minimum one, then select an ultra-low delay path not exceeding 5ms to progress PE2, assuming TE path-12 in Figure 3.

After receiving the second route advertisement, ABR knows that the suggested metric credit from itself to the downstream BGP speaker neighbor PE2 is 50ms, i.e., the average metric credit 50, the residual metric credit 100, taking the minimum one, then select a low delay path not exceeding 50ms to progress PE2, assuming TE path-22 in Figure 3.

ABR continues to advertise the above two BGP intent routes to the upstream BGP speaker neighbor PE1, where the metric is accumulated by the selected underlay path, the BGP next hop is modified to ABR, and the metric credit information is unchanged.

After receiving the first route advertisement, PE1 knows that the suggested metric credit from itself to the downstream BGP speaker neighbor ABR is 5ms, i.e., the average metric credit 5, the residual metric credit 6, taking the minimum one, then select an ultra-low delay path to ABR, assuming TE path-11 in Figure 3.

After receiving the second route advertisement, PE1 knows that the suggested metric credit from itself to the downstream BGP speaker neighbor ABR is 50ms, i.e., the average metric credit 50, the residual metric credit 60, taking the minimum one, then select a low delay path to ABR, assuming TE path-12 in Figure 3.

4.2. Example of Intent with Metric Credit Piece

Based on the first example, assume that two IGP domains are managed by a single provider, which is familiar with the performance of the network, and the propagated path of BGP intent route is clear, then, intent template may include explicit metric credit piece information.

Thus the intent routes originated from PE2 will contain more information related with metric credit piece.
The metric related information encoded in <prefix = loopback-PE2, intent-id = 1000> is:

metric-type: Unidirectional Link Delay
metric: assume an initial value of 0
metric-credit information:
  Count of Sources: 1
  S-Flag = 1, F-Flag = 0, P-Flag = 1
  Network Address of Source: loopback-PE1
  Total Metric Credit for Source: 10
  Estimated BGP Hops Count for Source: 2
  Current Hop Number: 0
  Metric Credit Piece [2]: [0] = 4, [1] = 6

Similarly, the metric related information encoded in <prefix = loopback-PE2, intent-id = 2000> is:

metric-type: Unidirectional Link Delay
metric: assume an initial value of 0
metric-credit information:
  Count of Sources: 1
  S-Flag = 1, F-Flag = 0, P-Flag = 1
  Network Address of Source: loopback-PE1
  Total Metric Credit for Source: 100
  Estimated BGP Hops Count for Source: 2
  Current Hop Number: 0
  Metric Credit Piece [2]: [0] = 40, [1] = 60

After receiving the first route advertisement, ABR knows that the suggested metric credit from itself to the downstream BGP speaker
neighbor ABR is 4ms, i.e., the explicit metric credit piece 4, the residual metric credit 10, taking the minimum one, then select an ultra-low delay path to PE2, assuming TE path-12 in Figure 3.

After receiving the second route advertisement, ABR knows that the suggested metric credit from itself to the downstream BGP speaker neighbor ABR is 40ms, i.e., the explicit metric credit piece 40, the residual metric credit 100, taking the minimum one, then select a low delay path to PE2, assuming TE path-22 in Figure 3.

ABR continues to advertise the above two BGP intent routes to the upstream BGP speaker neighbor PE1, where the metric is accumulated by the selected underlay path, the BGP next hop is modified to ABR, and the Current Hop Number is incremented by 1.

After receiving the first route advertisement, PE1 knows that the suggested metric credit from itself to the downstream BGP speaker neighbor ABR is 6ms, i.e., the explicit metric credit piece 6, the residual metric credit 6, taking the minimum one, then select an ultra-low delay path to ABR, assuming TE path-11 in Figure 3.

After receiving the second route advertisement, PE1 knows that the suggested metric credit from itself to the downstream BGP speaker neighbor ABR is 60ms, i.e., the explicit metric credit piece 60, the residual metric credit 60, taking the minimum one, then select a low delay path to ABR, assuming TE path-12 in Figure 3.

4.3. Example of Intent Shared by Multiple Path

As shown in Figure 4, it contains three AS. BGP neighbors are established between PE1 and ASBR1, ASBR1 and ASBR2, ASBR1 and ASBR3, ASBR2 and PE2, ASBR3 and PE3, to advertise BGP intent routes carrying intent identifier and metric credit attribute.
In this example, the same type of service needs to communicate between PE1 and PE2, PE1 and PE3. The intent of this type of service is the specific value related to the end-to-end total delay and distance of the transport path used.

PE2 and PE3 will respectively config the intent templates with the same intent identifier, e.g., intent-1000.

The intent-1000 template configured on PE2 have the following configuration:

```
metric-type: Unidirectional Link Delay (ms)
total-metric-credit: 200 (ms)
metric-credit enabled
```

The intent-1000 template also configured on PE3 have the following configuration:

```
metric-type: Unidirectional Link Delay (ms)
total-metric-credit: 300 (ms)
metric-credit enabled
```

PE2 advertises BGP intent route, <prefix = loopback-PE2, intent-id = 1000>, which is advertised to ASBR2. The metric related information encoded is:

```
metric-type: Unidirectional Link Delay
```
metric: assume an initial value of 0

metric-credit information:

Count of Sources: 1
S-Flag = 1, F-Flag = 0, P-Flag = 0
Network Address of Source: loopback-PE1
Total Metric Credit for Source: 200
Estimated BGP Hops Count for Source: 3

Similarly, PE3 advertises BGP intent route, <prefix = loopback-PE3, intent-id = 1000>, which is advertised to ASBR3. The metric related information encoded is:

metric-type: Unidirectional Link Delay
metric: assume an initial value of 0
metric-credit information:

Count of Sources: 1
S-Flag = 1, F-Flag = 0, P-Flag = 0
Network Address of Source: loopback-PE1
Total Metric Credit for Source: 300
Estimated BGP Hops Count for Source: 3

Then, ASBR2 and ASBR3 will use the suggested metric credit 66 and 100, taking minimum value from residual metric credit and average metric credit, to select the transport path for <prefix = loopback-PE2, intent-id = 1000> and <prefix = loopback-PE3, intent-id = 1000> respectively, assuming TE path-13 with cumulated metric 60 and TE path-23 with cumulated metric 100 in Figure 4.

Similarly, ASBR1 will also use suggested metric credit 66 and 100, to select the transport path for <prefix = loopback-PE2, intent-id = 1000> and <prefix = loopback-PE3, intent-id = 1000> respectively, assuming TE path-12 with cumulated metric 10 and TE path-22 with cumulated metric 10 in Figure 4.
Similarly, PE1 will also use suggested metric credit 66 and 100, to select the transport path for <prefix = loopback-PE2, intent-id = 1000> and <prefix = loopback-PE3, intent-id = 1000> respectively, assuming TE path-11 with cumulated metric 60 and TE path-21 with cumulated metric 100 in Figure 4.

5. IANA Considerations

This document request the METRIC_CREDIT entry to the "BGP Path Attributes" registry:

<table>
<thead>
<tr>
<th>Value</th>
<th>Code</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>METRIC_CREDIT</td>
<td>This Document</td>
</tr>
</tbody>
</table>

6. Security Considerations

TBD

7. Acknowledgements

TBD

8. Normative References

[I-D.dskc-bess-bgp-car]

[I-D.ietf-lsr-flex-algo-bw-con]

[I-D.kaliraj-idr-bgp-classful-transport-planes]


Authors’ Addresses

Peng & Tan

Expires July 1, 2022
Abstract

This document proposes extensions of BGP and defines some new Segment Types with algorithm information to meet more requirements when delivering SR Policy via 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 4 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1. Introduction

Segment Routing (SR) [RFC8402] allows a headend node to steer a packet flow along any path. [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 IPv6 data plane instantiations of Segment Routing with their respective representations of segments as SR-MPLS SID and SRv6 SID as described in [RFC8402].

[I-D.ietf-idr-segment-routing-te-policy] 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. It defines a new BGP address family (SAFI), i.e., SR Policy SAFI NLRI. In UPDATE messages of that address family, the NLRI identifies an SR Policy Candidate Path, and the attributes encode the segment lists and other details of that SR Policy Candidate Path. 11 Segment Types (from A to K) are defined to encode SR-MPLS or SRv6 segments.
As specified in [I-D.ietf-idr-segment-routing-te-policy], the SR algorithm can be optionally specified for Segment Types C(IPv4 Node and SID), D(IPv6 Node and SID for SR-MPLS), I(IPv6 Node and SID for SRv6), J(IPv6 Node, index for remote and local pair, and SID for SRv6), and K(IPv6 Local/Remote addresses and SID for SRv6). That is, currently the algorithm can be carried along with SR-MPLS prefix SID, SRv6 prefix SID and SRv6 adjacency SID when delivering SR Policy via BGP.

This document proposes extensions of BGP and defines some new Segment Types with algorithm information to meet more requirements when delivering SR Policy via BGP.

2. 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].

3. New Segment Types for SR-MPLS Adjacency with optional Algorithm

[I-D.ietf-lsr-algorithm-related-adjacency-sid] complements that besides Prefix-SID, the algorithm can be also included as part of an Adjacency-SID advertisement for SR-MPLS, in scenarios where multiple algorithm share the same link resource. In this case, an SR-MPLS Policy advertised to the headend may also contain algorithm specific Adjacency-SID.

This section defines 4 new Segment Sub-TLVs of Segment List Sub-TLV to provide algorithm information for SR-MPLS Adjacency-SID.

The processing procedures for SID with algorithm specified in [I-D.ietf-spring-segment-routing-policy] and [I-D.ietf-idr-segment-routing-te-policy] are still applicable for the new segment types. When the algorithm is not specified for the SID types above which optionally allow for it, the headend SHOULD use the Strict Shortest Path algorithm if available; otherwise, it SHOULD use the default Shortest Path algorithm.

3.1. Type M: IPv4 Address + Local Interface ID with optional Algorithm

The Type M Segment Sub-TLV is similar with existed Type E Segment Sub-TLV, it also encodes an IPv4 node address, a local interface Identifier (Local Interface ID) and an optional SR-MPLS SID, but with additional algorithm information. The format is as follows:
3.1. Type E: IPv4 Interface address with optional Algorithm

The Type E Segment Sub-TLV is similar with existed Type F Segment Sub-TLV, it also encodes an adjacency local address, an adjacency remote address and an optional SR-MPLS SID, but with additional algorithm information. 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 | SR Algorithm |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Local Interface ID (4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IPv4 Node Address (4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SR-MPLS SID (optional, 4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

Type: TBD1

SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] 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 set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as the existing Type E Segment Sub-TLV.

3.2. Type N: IPv4 Local and Remote addresses with optional Algorithm

The Type N Segment Sub-TLV is similar with existed Type F Segment Sub-TLV, it also encodes an adjacency local address, an adjacency remote address and an optional SR-MPLS SID, but with additional algorithm information. 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 | SR Algorithm |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Local IPv4 Address (4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Remote IPv4 Address (4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| SR-MPLS SID (optional, 4 octets) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:
Type: TBD2

SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] 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 set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as existed Type F Segment Sub-TLV.

3.3. Type O: IPv6 Address + Interface ID for local and remote pair with optional Algorithm related SID for SR MPLS

The Type O Segment Sub-TLV is similar with existed Type G Segment Sub-TLV, it also 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 SR-MPLS SID, but with additional algorithm information. The format is as follows:

```
  0  1  2  3
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
| Type  | Length | Flags | SR Algorithm | Local Interface ID (4 octets) | IPv6 Local Node Address (16 octets) | Remote Interface ID (4 octets) | IPv6 Remote Node Address (16 octets) | SR-MPLS SID (optional, 4 octets) |
+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+-------------------------------+
```

Where:

Type: TBD3
SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] 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 set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as existed Type G Segment Sub-TLV.

3.4. Type P: IPv6 Local and Remote addresses with optional Algorithm for SR MPLS

The Type P Segment Sub-TLV is similar with existed Type H Segment Sub-TLV, it also encodes an adjacency local address, an adjacency remote address and an optional SR-MPLS SID, but with additional algorithm information. The format is as follows:

```
+----------------+----------------+----------------+----------------+----------------+----------------+----------------+----------------+
| Type           | Length        | Flags         | SR Algorithm   |
+----------------+----------------+----------------+----------------+
| Local IPv6 Address (16 octets) |
+----------------+----------------+----------------+----------------+
| Remote IPv6 Address (16 octets) |
+----------------+----------------+----------------+----------------+
| SR-MPLS SID (optional, 4 octets) |
```

Where:

Type: TBD4

SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] 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 set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as existed Type H Segment Sub-TLV.
4. New Segment Types for SID only, with optional Algorithm

Segment Sub-TLV for Type A defined in section 2.4.4.2.1 [I-D.ietf-idr-segment-routing-te-policy] carries only the SID information in the form of MPLS Label. Segment Sub-TLV for Type B defined in section 2.4.4.2.2 [I-D.ietf-idr-segment-routing-te-policy] carries only the SID information in the form of IPv6 address.

If the algorithm information is carried along with the SIDs, it’s useful in the scenarios below:

Scenario 1: The algorithm may be optionally provided to the headend for verification purposes. The headend can check if the SID value and the related algorithm received can be found in its SR-DB if requested to do so.

Scenario 2: The headend may not know about the SID-related algorithm especially in the inter-domain scenario. Providing the algorithm information benefits troubleshooting and network management.

This section defines 2 new Segment Sub-TLVs of Segment List Sub-TLV to provide algorithm information for SR-MPLS/SRv6 SID.

4.1. Type L: MPLS SID only, with optional Algorithm

The Type L Segment Sub-TLV is similar with the Type A Segment Sub-TLV, it also encodes a single SR-MPLS SID, but with additional algorithm information. 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     |  SR Algorithm |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Label                        | TC  |S|       TTL     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Where:

Type: TBD5

SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] is present. When A-Flag is not encoded, this field SHOULD be set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as Type A Segment Sub-TLV.
4.2. Type Q: SRv6 SID only, with optional Algorithm

The Type Q Segment Sub-TLV is similar with existed Type B Segment Sub-TLV, it also encodes a single SRv6 SID, but with additional algorithm, endpoint behavior and SID structure information. 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     |  SR Algorithm |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                       SRv6 SID (16 octets)                  //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//     SRv6 Endpoint Behavior and SID Structure (optional)     //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

Where:

Type: TBD6

Length is variable.

SR Algorithm: 1 octet specifying SR Algorithm as described in section 3.1.1 in [RFC8402] when A-Flag as defined in section 2.4.4.2.12 [I-D.ietf-idr-segment-routing-te-policy] is present. When A-Flag is not encoded, this field SHOULD be set to zero on transmission and MUST be ignored on receipt.

Other fields have the same meaning as the Type B Segment Sub-TLV.

5. IANA Considerations

This document requests codepoint allocations for new Segment Sub-TLVs in the "SR Policy List Sub-TLVs" registry.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>Type L MPLS Algorithm related SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD2</td>
<td>Type M IPv4 Node, index and Algorithm related SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD3</td>
<td>Type N IPv4 Local/Remote addresses and Algorithm related SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD4</td>
<td>Type O IPv6 Node, index for remote and local pair and Algorithm related SID for SR-MPLS sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD5</td>
<td>Type P IPv6 Local/Remote addresses and Algorithm related SID sub-TLV</td>
<td>This document</td>
</tr>
<tr>
<td>TBD6</td>
<td>Type Q SRv6 Algorithm related SID sub-TLV</td>
<td>This document</td>
</tr>
</tbody>
</table>
6. Security Considerations

Procedures and protocol extensions defined in this document do not affect the security considerations discussed in [I-D.ietf-idr-segment-routing-te-policy].

7. References

7.1. Normative References

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

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


7.2. Informative References

[I-D.ietf-lsr-algorithm-related-adjacency-sid]

[I-D.ietf-lsr-flex-algo]

Liu & Peng Expires 4 September 2022
[I-D.ietf-lsr-isis-srv6-extensions]

[I-D.ietf-lsr-ospfv3-srv6-extensions]


Authors’ Addresses

Yao Liu
ZTE
Nanjing
China
Email: liu.yao71@zte.com.cn

Shaofu Peng
ZTE
Nanjing
China
Email: peng.shaofu@zte.com.cn
BGP Flowspec Redirect Load Balancing Group Community
draft-wu-idr-flowspec-redirect-group-00

Abstract

This document defines an extension to "BGP Community Container Attribute" [I-D.ietf-idr-wide-bgp-communities], which allows flowspec redirection to multiple paths. This extended community serves to redirect traffic to a load balancing group and supports both equal-cost multi-path (ECMP) and unequal-cost multi-path (UCMP) scenarios.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

"Redirect to IP Extended Community", defined in
[I-D.ietf-idr-flowspec-redirect-ip], allows traffic to be redirected
to a specific IPv4 or IPv6 address, and
[I-D.jiang-idr-ts-flowspec-srv6-policy] defines the redirection
action to a SRv6 tunnel by additionally carrying the "Color Extended
Community" [RFC8955].
However, scenarios involving redirection load balancing are not described in both documents. Although in some implementations, Equal-cost multi-path (ECMP) of "Redirect to IP" action can be achieved by encoding multiple redirect Extended Communities, the current set of mechanisms can hardly support neither ECMP of SRv6 tunnels nor unequal-cost multi-path (UCMP) of either types.

This document defines an extension to "BGP Community Container Attribute" [I-D.ietf-idr-wide-bgp-communities], the "Redirect Load Balancing Group" community. It is a new type of wide community container attribute with encoding format of multiple redirection path TLVs. Each of these TLVs represents a different redirection action. It allows traffic redirection to a load balancing group and supports both ECMP and UCMP scenarios.

1.1. Terminology

This document introduces the following terms:

ECMP: Equal-Cost Multi-Path

UCMP: Unequal-Cost Multi-Path

2. Redirect Load Balancing Group Community

This document defines a new type of "BGP Community Container Attribute", the "Redirect Load Balancing Group" community type. The format complies with "BGP Community Container Attribute" [I-D.ietf-idr-wide-bgp-communities] and is shown below:
The Type, Flags, Reserved and Length fields comply with the "BGP Community Container Attribute Common Header" definition.

The container type MUST be 1, which represents BGP Wide Community.

The Length field represents the total length of the container’s contents in octets.

2.1. Community Value

The Community Value, Source AS Number and Context AS Number fields comply with the corresponding definition in "BGP Community Container Attribute".

Community Value: 4 octets value that represents the "Redirect Load Balancing Group" community type. The value is TBD and requires IANA registration; See Section 5.1.

2.2. Param TLV

The BGP Wide Community Parameter TLV (Sub-Type 3) contains a list of atoms, comply with "BGP Wide Community Parameter(s) TLV" section of "BGP Community Container Attribute".

Figure 1: Redirect Load Balancing Group Community Format

Wu, et al.          Expires 8 September 2022
The Parameter TLV MUST present and SHOULD appear only once in a "Redirect Load Balancing Group" community container, no or multiple present SHOULD be considered malformed.

Sub-Type: Type 3 (BGP Wide Community Parameter TLV)

Length: Length of all the sub-TLVs in octets.

2.3. Sub-TLVs (Atoms)

The list of atoms that Param Tlv contains. Each atom represents a different redirection path.

The general format of the sub-TLVs comply with atoms’ format defined in "BGP Community Container Attribute", as below:

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

Figure 2: Param Sub-TLV Format

The Type field is an octet from 1~254 (0 and 255 are reserved). Supported type of the sub-TLVs includes:

Type 1: IPv4 Prefix Only
Type 2: IPv4 Prefix with Weight
Type 3: IPv4 Prefix with Color
Type 4: IPv4 Prefix with Color and Weight
Type 5: IPv6 Prefix Only
Type 6: IPv6 Prefix with Weight
Type 7: IPv6 Prefix with Color
Type 8: IPv6 Prefix with Color and Weight
These sub-TLV types SHOULD be used exclusively within "Redirect Load Balancing Group" community containers.

The Length represents the length of the "Value" field in octets, and it is fixed for each specific sub-TLV.

If the length and type of a sub-TLV do not match, the "Redirect Load Balancing Group" community container SHOULD be considered malformed.

If a sub-TLV is a total duplication of a previous one, the latter sub-TLV MUST be ignored.

In principle, sub-TLVs of different types may be combined in any mode. The supported combinations depend on the specific implementation.

2.3.1. Atom Type 1: IPv4 Prefix Only

Indicating the redirection path is unweighted and to a IPv4 address. The format is shown below:

```
| Type: 1 | Length: 6 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flag(2) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IPv4(4) |
```

**Figure 3: Atom Type 1: IPv4 Prefix Only**

Length: MUST be 6.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the reciever.

IPv4: 4-octet IPv4 address, redirection destination

2.3.2. Atom Type 2: IPv4 Prefix with Weight

Indicating the redirection path is weighted and to a IPv4 address. The format is shown below:
2.3.3. Atom Type 3: IPv4 Prefix with Color

Indicating the redirection path is unweighted and to a SR-TE tunnel. The format is shown below:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Type: 3 | Length: 10 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Flag(2) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| IPv4(4) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Color(4) |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Atom Type 3: IPv4 Prefix with Color

Length: MUST be 10.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv4: 4-octet IPv4 address, SR-TE tunnel Endpoint for redirection
2.3.4. Atom Type 4: IPv4 Prefix with Color and Weight

Indicating the redirection path is weighted and to a SR-TE tunnel. The format is shown below:

```
+---------------+---------------+---------------+---------------+
|   Type: 4     |   Length: 11  |
|               |               |
|               | Flag(2)       |
|               | IPv4(4)       |
|               | Color(4)      |
|               | Weight(1)     |
```

Figure 6: Atom Type 4: IPv4 Prefix with Color and Weight

Length: MUST be 11.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv4: 4-octet IPv4 address, SR-TE tunnel Endpoint for redirection

Color: 4 octets, SR-TE tunnel Color for redirection

Weight: 1 octet, values from 1-255, load balancing weight

2.3.5. Atom Type 5: IPv6 Prefix Only

Indicating the redirection path is unweighted and to a IPv6 address. The format is shown below:
Figure 7: Atom Type 5: IPv6 Prefix Only

Length: MUST be 18.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv6: 16-octet IPv6 address, redirection destination

2.3.6. Atom Type 6: IPv6 Prefix with Weight

Indicating the redirection path is weighted and to a IPv6 address. The format is shown below:

Length: MUST be 19.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv6: 16-octet IPv6 address, redirection destination

Weight: 1 octet, values from 1-255, load balancing weight
2.3.7. Atom Type 7: IPv6 Prefix with Color

Indicating the redirection path is unweighted and to a SRv6 tunnel. The format is shown below:

```
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
|                  |                  |                  |                  |
|                  |                  |                  |                  |
| Type: 7          | Length: 22      |                  |                  |
|                  |                  |                  |                  |
| Flag(2)          | IPv6(16)        |                  |                  |
|                  |                  |                  |                  |
| Color(4)         |                  |                  |                  |
```

Length: MUST be 22.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv6: 16-octet IPv6 address, SRv6 tunnel Endpoint for redirection

Color: 4 octets, SRv6 tunnel Color for redirection

2.3.8. Atom Type 8: IPv6 Prefix with Color and Weight

Indicating the redirection path is weighted and to a SRv6 tunnel. The format is shown below:

```
+---------------+---------------+---------------+---------------+---------------+---------------+---------------+---------------+
|                  |                  |                  |                  |
|                  |                  |                  |                  |
| Type: 8          | Length: 23      |                  |                  |
|                  |                  |                  |                  |
| Flag(2)          | IPv6(16)        |                  |                  |
|                  |                  |                  |                  |
| Color(4)         |                  |                  |                  |
| Weight(1)        |                  |                  |                  |
```

Figure 9: Atom Type 7: IPv6 Prefix with Color

Figure 9: Atom Type 8: IPv6 Prefix with Color and Weight
Figure 10: Atom Type 8: IPv6 Prefix with Color and Weight

Length: MUST be 23.

Flags: 2 octets, reserved for future use, MUST be set to 0 upon the sender and MUST be ignored upon the receiver.

IPv6: 16-octet IPv6 address, SRv6 tunnel Endpoint for redirection

Color: 4 octets, SRv6 tunnel Color for redirection

Weight: 1 octet, values from 1~255, load balancing weight

3. Scenarios

This section describes a few use-case scenarios when deploying "Redirect Load Balancing Group" community type.

Weighted atom types: Atoms contain a Weight field, such as Type 2, 4, 6, 8

Unweighted atom types: Atoms do not contain a Weight field, such as Type 1, 3, 5, 7

3.1. ECMP

A system that originates a flowspec route with a "Redirect Load Balancing Group" community, among which its parameter TLV contains more than 1 atoms. If not all atoms are of a weighted type, these atoms will form a ECMP group.

Implementations MUST be prepared to accept a Parameter TLV with both weighted and unweighted atoms. In this case, the Weight field of the weighted atom SHOULD be ignored.

3.2. UCMP

A system that originates a flowspec route with a "Redirect Load Balancing Group" community, among which its parameter TLV contains more than 1 atoms. If all atoms are of a weighted type, these atoms will form a UCMP group.

In this case, the Weight field value of these atoms SHOULD NOT be ignored, and the values are used as the ratio of the UCMP group.
4. Error Handling

Comply with Error Handling Procedure in "BGP Community Container Attribute" [I-D.ietf-idr-wide-bgp-communities].

In addition:

4.1. Redirect Group Wide Community Parameter TLV

A "Redirect Load Balancing Group" community container with no or multiple parameter TLVs SHOULD be considered malformed, and a "treat as withdraw" behavior is expected.

4.2. Redirect Group Wide Community Parameter Sub-TLVs

If the length and type of a sub-TLV do not match, the "Redirect Load Balancing Group" community container SHOULD be considered malformed, and a "treat as withdraw" behavior is expected.

5. IANA Considerations

5.1. BGP Wide Communities Community Type : Redirect Group

This document requests a new community value under "Registered Type 1 BGP Wide Community Community Types" registry. This registry is defined and requested in "BGP Community Container Attribute" [I-D.ietf-idr-wide-bgp-communities].

Requested value:

<table>
<thead>
<tr>
<th>Name</th>
<th>Type Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redirect Load Balancing Group</td>
<td>TBD</td>
</tr>
</tbody>
</table>

6. Security Considerations

A system that originates a flowspec route with a "Redirect Load Balancing Group" BGP wide community can cause many receivers of that route to redirect traffic to a single next-hop, overwhelming that next-hop and resulting in inadvertent or deliberate denial-of-service. This is also a concern about the "redirect to IP" extended community, therefore this document introduces no additional security considerations than those already covered in [RFC8955].

7. References

7.1. Normative References

Wu, et al. Expires 8 September 2022
7.2. References


Authors’ Addresses

Zhiwen Wu
Huawei Technologies
No. 156 Beiqing Road
Beijing
100095
P.R. China
Email: wuzhiwen1@huawei.com

Haibo Wang
Huawei Technologies
No. 156 Beiqing Road
Beijing
100095
P.R. China
Email: rainsword.wang@huawei.com

Lili Wang
Huawei Technologies
No. 156 Beiqing Road
Beijing
100095
P.R. China
Email: lily.wong@huawei.com

Zhen Tan
Huawei Technologies
No. 156 Beiqing Road
Beijing
100095
P.R. China
Email: tanzhen6@huawei.com
Abstract

This document proposes extensions to BGP Flow Specification for the flow mapping of Deterministic Networking (DetNet) when interconnected with IEEE 802.1 Time-Sensitive Networking (TSN). The BGP flowspec is used for the filtering of the packets that match the DetNet networks and the mapping between TSN streams and DetNet flows in the control plane.

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 2 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

[RFC8655] specifies the architecture of Deterministic Networking (DetNet), which provide a capability for the delivery of data flows with extremely low packet loss rates and bounded end-to-end delivery latency. DetNet-enabled end systems and DetNet nodes can be interconnected by sub-networks, i.e., Layer 2 technologies such as IEEE 802.1 Time-Sensitive Networking (TSN).

As defined in [RFC8655], the DetNet IP and MPLS flows can be carried over TSN sub-networks. DetNet needs to be mapped to the sub-networks technology used to interconnect DetNet nodes. For example, a TSN node may be used to interconnect DetNet-aware nodes, and these DetNet nodes can map DetNet flows to TSN streams. When the Detnet provide the deterministic service for the TSN end system, a DetNet edge node may be used to interconnect the TSN end system, and the DetNet nodes can map the TSN streams to DetNet flows.
As described in [RFC8938], one of the primary requirements of the DetNet Controller Plane is restricting flows to IEEE 802.1 TSN and the requirement could use the centralized network management provisioning mechanisms such as BGP protocol. As defined in [RFC8955], the Flow Specifications for BGP is an n-tuple consisting of several matching criteria which is comprised of traffic filtering rules and is associated with actions that can be applied to the traffic flows. The DetNet edge nodes can provide the capability to process the traffic including classifying, shaping, rate limiting, filtering, and redirecting packets based on the policies configured by the BGP Flow Specification.

BGP flow specification version 1 (FSv1) has been defined in [RFC8955] and version 2 of the BGP flow specification (FSv2) protocol has been proposed in [I-D.hares-idr-flowspec-v2]. This document proposes extensions to BGP FSv2 for the interconnection of DetNet and TSN. The BGP flowspec is used for the filtering of the packets that match the DetNet networks and the mapping between TSN streams and DetNet flows in the control plane.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655], [RFC8938], [RFC8955] and [I-D.hares-idr-flowspec-v2].

2.2. 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.

3. The Requirements for DetNet Control Plane

3.1. Functions for DetNet Flow to TSN Stream Mapping

As described in [RFC9024], TSN networks can be interconnected over a DetNet MPLS Network. And as discussed in [RFC9023] and [RFC9037], DetNet IP or MPLS networks can be operating over a TSN sub-network. The mapping between TSN Streams and DetNet flows is required for the service proxy function at DetNet Edge nodes. And the mapping table can be configured and maintained in the control plane. When a DetNet Edge Node receives a packet, it MUST identify and check whether such flow is present in its mapping table and decide to drop (when not match) or to forward the packet (when match) to the associated
As Figure 1 shows, it is required to configure the identification information when mapping received TSN Streams to the DetNet flows at Edge Node-1. Mechanisms and Parameters of TSN stream identification (e.g., Mask-and-Match Stream identification) defined in [IEEE8021CB] and [IEEE8021CBdb] can be used for service proxy function. After the identification of the TSN stream, it needs to map the packet to the DetNet flow information such as S-Label, d-CW when in DetNet MPLS data plane and handle the packet as defined in [RFC8964].

When the DetNet Edge Node-2 receives a DetNet flow, it MUST identify the DetNet flow-ID information such as IP 6-tuple in DetNet IP data plane or S-Label and d-CW information in DetNet MPLS data plane. Then the Service proxy function needs to map the DetNet flow-ID and flow related parameters to the associated TSN Stream IDs and streams related parameters.

Flow Mapping:

<table>
<thead>
<tr>
<th>TSN</th>
<th>DetNet</th>
<th>Edge Node-1</th>
<th>Transit Node</th>
<th>Edge Node-2</th>
<th>DetNet</th>
<th>TSN End System</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSN</td>
<td>+++++--- End to End TSN Service +++++--</td>
<td>TSN</td>
<td>+++++---</td>
<td>TSN</td>
<td>+++++---</td>
<td>TSN</td>
</tr>
<tr>
<td>Applic.</td>
<td>+++++---</td>
<td>Service-Proxy</td>
<td>Service-Proxy</td>
<td>Service-Proxy</td>
<td>Service-Proxy</td>
<td>Applic.</td>
</tr>
<tr>
<td>TSN</td>
<td>++----&lt;-- DetNet flow --++----.+</td>
<td>TSN</td>
<td>++----&lt;-- DetNet flow --++----.+</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSN</td>
<td></td>
<td>TSN</td>
<td>Svc</td>
<td></td>
<td>TSN</td>
<td>Svc</td>
</tr>
<tr>
<td>L2</td>
<td>L2</td>
<td>Fwd</td>
<td>Forwarding</td>
<td>Fwd</td>
<td>L2</td>
<td>L2</td>
</tr>
<tr>
<td>L2</td>
<td>L2</td>
<td>Fwd</td>
<td>Forwarding</td>
<td>Fwd</td>
<td>L2</td>
<td>L2</td>
</tr>
</tbody>
</table>

Figure 1: Flow Mapping in TSN over DetNet Network
3.2. Aggregation during DetNet Flow to TSN Stream Mapping

As described in [RFC8938], the DetNet data plane allows for the aggregation of DetNet flows, which should also be accomplished in the control plane. IP, MPLS and TSN aggregation has both data plane and controller plane aspects. Bandwidth reservations, resource assignment, path computation, delay, delay variation and aggregate number should be taken into considerations in the controller plane. Moreover, as defined in [RFC9023] and [RFC9037], 1:1 and N:1 mapping (aggregating multiple TSN Streams in a single DetNet flow) MUST be supported.

4. BGP Extensions for Flow Specification Encoding

As defined in [RFC8955], the nodes that applied a Flow Specification can filter the received packets according to the matching criteria and can forward the flows based on the associated actions. This document proposes extensions to BGP Flow Specification for the mapping of DetNet flows and TSN streams by using the traffic filtering rules to identify the packet and using the associated action to map the packet to the related service.

4.1. Filtering Rules for TSN Streams

As IEEE Std 802.1Q defined, a Stream ID is a 64-bit field that uniquely identifies a stream and can be generated by the system offering the stream, or possibly a device controlling that system. But it is not carried in the header of the TSN Stream. As defined in [IEEE8021CB] and [IEEEP8021CBdb], five specific Stream identification functions are described: Null Stream identification, Source MAC and VLAN Stream identification, Active Destination MAC and VLAN Stream identification, and IP Stream identification, and Mask-and-match Stream identification. It needs to examine the header of the streams such as destination_address, vlan_identifier, IP source address, IP destination address, DSCP, IP next protocol, source port, destination port and mac_service_data_unit.

As defined in [I-D.ietf-idr-flowspec-l2vpn], the Ethernet Layer 2 (L2) related fields have been covered by the L2 traffic filtering rules except the mac_service_data_unit in Mask-and-Match Stream identification. A mac_service_data_unit mask is defined to identify communication flows supported by various higher-layer protocols. L2 Traffic Rules and L2 header TLV in BGP FSv2 of has been defined in [I-D.hares-idr-flowspec-v2] section 3.4. This document proposes a new L2 SubTLV for TSN Streams in L2 Flow Specification Component shown in Figure 2.
SubTLV type = TBD1: Mac Service Data Unit

Encoding: <type (1 octet), length (1 octet), [op, value]+>

Defines a list of {operation, value} pairs used to match 6-octet Mac Service Data Unit field. Values are encoded as 6-octet quantities. op is encoded as specified in Section 4.2.1.1 of [RFC8955].

4.2. Traffic Action for TSN Streams

The action for an TSN traffic filtering flowspec is to accept the TSN streams that matches that particular rule and map the streams to the DetNet flows. The action for L3 traffic with extended communities types per [RFC8955] and [RFC8956] such as traffic-rate, traffic-marking, traffic-action, and redirect can be used for TSN to DetNet IP flow mapping. The Wide Community has been proposed for FSv2 actions in [I-D.hares-idr-flowspec-v2] section 3.2.

The DetNet flow is identified by a S-Label and the DetNet Header consists of d-CW and F-Labels. The MPLS label related action for an TSN stream mapping to a DetNet MPLS network can use the Label-action defined in [I-D.ietf-idr-bgp-flowspec-label]. And the action for the sequence in d-CW field, this document proposes a new Action SubTLV in BGP FSv2 Wide Community for TSN Streams as following shown.

```
+======+=================+==========+
| type | Wide Community  | encoding |
+======+=================+==========+
| TBD2 | Sequence Action | bitmask  |
+========================================+
```

Table 1

The The Sequence Action SubTLV is shown in Figure 3.
Figure 3: Sequence Action

Type: 4 bits, indicates the length type of the sequence number:

0: 0 bits
1: 16 bits
2: 28 bits

Sequence Number: 28 bits, an unsigned value implementing the DetNet sequence number.

4.3. Filtering Rules for DetNet Flows

The L3 traffic filtering rules defined in [RFC8955] and [RFC8956] can be used for DetNet IP flow.

As defined in RFC8964, the MPLS-based DetNet data plane encapsulation consists of d-CW, S-Label and F-Labels. The MPLS label filtering rules have been defined in [I-D.ietf-idr-flowspec-mpls-match]. IP header TLV in BGP FSv2 of has been defined in [I-D.hares-idr-flowspec-v2] section 3.1.

This document proposes a new IP header SubTLV for DetNet MPLS flows shown in Figure 4.
MPLS Match Type TBD3: d-CW, indicates Sequence in Label stack.

Encoding: <type (1 octet), length (1 octet), [op, value]+>

Defines a list of {operation, value} pairs used to match Sequence. Values are encoded as 4-octet quantities, where the four most significant bits are set to zero and ignored for matching and the 28 least significant bits contain the sequence value. op is encoded as specified in Section 4.2.1.1 of [RFC8955].

4.4. Traffic Action for DetNet Flows

The extended action for an DetNet traffic filtering flowspec is to accept the DetNet flows that matches that particular rule and map the flows to the TSN streams. This document proposes a new Action SubTLV in BGP FSv2 Wide Community for DetNet flows as the following shown.

\[+-------------------+
| type | Wide Community | encoding |
+-------------------+
| TBD4 | TSN Action     | bitmask  |
+-------------------+

Table 2

The TSN Action SubTLV is shown in Figure 3.
Figure 5: TSN Action

Type: 1-octet, indicates the type of TSN profiles. The value of the types is TBD:

Resv: 1-octet, reserved for future use. MUST be sent as zero and ignored on receipt.

TSN-profile: 4-octet, can be converted to the TSN Stream ID and stream related parameters and requirements as the following shown.

stream_handle: identifying the Stream to which the packet belongs in TSN networks.

sequence_number: identifying the order in which the packet was transmitted relative to other packets in the same Compound Stream in TSN networks.

traffic_scheduling: identifying the traffic scheduling mechanisms including traffic policy, queuing and forwarding methods in TSN networks.

5. Security Considerations

TBA

6. Acknowledgements

TBA

7. IANA Considerations

TBA
8. Normative References

[I-D.hares-idr-flowspec-v2]

[I-D.ietf-idr-bgp-flowspec-label]

[I-D.ietf-idr-flowspec-l2vpn]

[I-D.ietf-idr-flowspec-mpls-match]


Authors’ Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan
Hubei, 430223
China
Email: xiong.quan@zte.com.cn
Haisheng Wu
ZTE Corporation
Nanjing
Jiangsu,
China
Email: wu.haisheng@zte.com.cn

Junfeng Zhao
CAICT
China
Email: zhaojunfeng@caict.ac.cn
Layer-3 Neighbor Discovery

draft-ymbk-idr-l3nd-04

Abstract

Data Centers where the topology is BGP-based need to discover neighbor IP addressing, IP Layer-3 BGP neighbors, etc. This Layer-3 Neighbor Discovery protocol identifies BGP neighbor candidates.

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 6 October 2022.
Copyright Notice

Copyright (c) 2022 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
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 Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction .............................................. 3
2. Terminology ............................................ 4
3. Background .............................................. 5
4. Inter-Link Protocol Overview .............................. 5
   4.1. L3ND Ladder Diagram ................................ 6
5. TLV PDUs ................................................ 7
6. HELLO .................................................... 8
   6.1. Transport ........................................... 9
   6.2. Flags ................................................. 10
   6.3. Port ................................................ 10
7. TCP Set-Up ............................................... 11
8. OPEN ...................................................... 12
9. ACK ....................................................... 14
   9.1. Retransmission ..................................... 15
10. The Encapsulations ...................................... 15
   10.1. The Encapsulation PDU Skeleton .................... 16
   10.2. Encapsulation Flags ................................ 17
   10.3. IPv4 Encapsulation ................................ 18
   10.4. IPv6 Encapsulation ................................ 18
   10.5. MPLS Label List .................................. 19
   10.6. MPLS IPv4 Encapsulation ......................... 20
   10.7. MPLS IPv6 Encapsulation ......................... 20
11. VENDOR - Vendor Extensions ............................. 21
12. Discussion ............................................... 22
   12.1. HELLO Discussion ................................ 22
13. VLANs/SVIs/Sub-interfaces ............................... 22
14. Implementation Considerations ........................... 22
15. Security Considerations ................................ 23
16. IANA Considerations ................................... 23
   16.1. Link Local Layer-3 Addresses ................... 24
   16.2. Ports for TLS/TCP ............................... 24
   16.3. PDU Types ....................................... 24
1. Introduction

The Massive Data Center (MDC) environment presents unusual problems of scale, e.g. O(10,000) forwarding devices, while its homogeneity presents opportunities for simple approaches. Layer-3 Discovery and Liveness (L3DL), [I-D.ietf-lsvr-l3dl], provides neighbor discovery at Layer-2. This document (set) provides a similar solution at Layer-3, attempting to be as similar as reasonable to L3DL.

Some guiding principles when dealing with datacenters with tens of thousands of devices are

* Predictable Reliability,

* Security: Authorization and Integrity more than Confidentiality, and

* Massive Scalability

Layer-3 Neighbor Discovery (L3ND) provides brutally simple mechanisms for neighboring devices to

* Discover each other’s IP Addresses,

* Discover mutually supported layer-3 encapsulations, e.g. IPv4/IPv6/MPLS,

* Discover Layer-3 IP and/or MPLS addressing of interfaces of the encapsulations,

* Provide authenticity, integrity, and verification of protocol messages, and

* Accommodate scaling needed for EVPN etc.
L3ND is intended for use within single IP subnets (IP over Ethernet or other point-to-point or multi-point IP link) in order to exchange the data needed to bootstrap BGP-based peering, EVPN, etc.; especially in a datacenter Clos [Clos] environment. Once IP connectivity has been leveraged to discover layer-3 addressability and forwarding capabilities, normal IP forwarding and routing can take over.

L3ND might be more widely applicable to a range of routing and similar protocols which need Layer-3 neighbor discovery.

2. Terminology

Even though it concentrates on the inter-device layer, this document relies heavily on routing terminology. The following attempts to clarify the use of some possibly confusing terms:

Clos: A hierarchic subset of a crossbar switch topology commonly used in data centers [Clos].

Datagram: The L3ND content of a single Layer-3 UDP Datagram.

Encapsulation: Address Family Indicator and Subsequent Address Family Indicator (AFI/SAFI). I.e. classes of Layer-2.5 and Layer-3 addresses such as IPv4, IPv6, MPLS.

Link or Logical Link: A logical connection between two interfaces on two different devices. E.g. two VLANs between the same two ports are two links.

MDC: Massive Data Center, commonly composed of thousands of Top of Rack Switches (TORs).

MTU: Maximum Transmission Unit, the size in octets of the largest packet that can be sent on a medium, see [RFC1122] 1.3.3.

PDU: Protocol Data Unit, an L3ND application layer message.

Session: An established, via exchange of OPEN PDUs, session between two L3ND capable IP interfaces on a link.

TOR Switch: Top Of Rack switch, aggregates the servers in a rack and connects to aggregation layers of the Clos tree, AKA the Clos spine.
3. Background

L3ND is primarily designed for a Clos type datacenter scale and topology, but can accommodate richer topologies which contain potential cycles.

While L3ND is designed for the MDC, there are no inherent reasons it could not run on a WAN. The authentication and authorization needed to run safely on a WAN need to be considered, and the appropriate level of security options chosen.

The number of addresses of one Encapsulation type on an interface link may be quite large given a TOR switch with tens of servers, each server having a few hundred micro-services, resulting in an inordinate number of addresses. And highly automated micro-service migration can cause serious address prefix disaggregation, resulting in interfaces with thousands of disaggregated prefixes.

To meet such scaling needs, the L3ND protocol is session oriented and uses incremental announcement and withdrawal with session restart, a la BGP ([RFC4271]).

4. Inter-Link Protocol Overview

A device broadcasts a Layer-3 Multicast UDP datagram (HELLO) containing the port number that is willing to serve a TLS or raw TCP connection to support the data exchange of the rest of the protocol in a reliable and preferably authenticated manner.

Another device on the link then establishes a TLS or raw TCP session in which inter-device PDUs are used to exchange device and logical link identities and layer-2.5 (MFLS) and 3 identifiers (not payloads), e.g. more IP Addresses, loopback addresses, port identities, and Encapsulations.

To assure discovery of new devices coming up on a multi-link topology, devices on such a topology, and only on a multi-link topology, send periodic HELLOs forever, see Section 12.1.

Given the TLS/TCP session, OPEN PDUs (Section 8) are exchanged, the Encapsulations (Section 10) configured on an end point may be announced and modified. Note that these are only the encapsulation and addresses configured on the announcing interface; though a device’s loopback and overlay interface(s) may also be announced.
4.1. L3ND Ladder Diagram

The HELLO, Section 6, is a priming message sent on all logical links configured for L3ND. It is a small L3ND Multicast UDP PDU with the simple goal of advertising a TLS/TCP service available on an advertised port on the sending IP interface.

The HELLO PDU is either IPv4 or IPv6, which selects the AFI to be used for the rest of the session(s) between end-points. Two endpoints MAY establish a link for each AFI.

An interface on the link receiving the HELLO PDU attempts to establish a TLS or raw TCP, as specified by the HELLO, session to the source IP address of the HELLO on the port advertised in the HELLO.

The OPEN, Section 8 PDUs, used to exchange details about the L3ND session, and the ACK/ERROR PDU, are mandatory; other PDUs are optional; though at least one encapsulation SHOULD be agreed at some point.

Like Multi-Protocol BGP, [RFC4760], an L3ND session running over one AFI MAY carry encapsulations etc. of different AFIs.

A L3DL extension, [I-D.ymbk-idr-l3nd-ulpc], describes the next upper layer L3DL protocol to exchange BGP parameter information.

The following is a ladder-style diagram of the L3ND protocol exchanges:

```
| HELLO -------------> Logical Link Peer discovery |
    | TCP OPEN               | Mandatory |
    | OPEN -------------> IDs, security, etc. |
    | ACK               |
    | OPEN -------------> Mandatory |
    | ACK               |
    | Interface IPv4 Addresses -------------> Optional |
```

Bush, et al. Expires 6 October 2022
5. TLV PDUs

The basic L3ND application layer PDU is a typical TLV (Type Length Value) PDU. As it is transported over TCP, integrity is assured. When it is transported over TLS, authenticity is also provided.
The fields of the basic L3ND header are as follows:

Version: An integer differentiating versions of the L3ND protocol. Currently only Version 0 MAY BE specified.

PDU Type: An integer differentiating PDU payload types. See Section 16.3.

Payload Length: Total number of octets in the Payload field.

Payload: The application layer content of the L3ND PDU.

6. HELLO

The Payload Length is 4 to cover the Transport, Flags, and Port fields.

The IPv4 UDP packets are sent to the IPv4 link local multicast address (TBD1) and the IPv6 UDP packets are sent to an IPv6 link Local multicast address (TBD2). See Section 12.1 for why multicast is used.

The HELLO PDU solicits a unicast TLS/TCP session open request of the same AFI from other devices on the link.
When a HELLO is received from a source IP address with which there is no established TLS/TCP L3ND session, if the receiver has the higher of the two IP addresses, it SHOULD respond by sending a TLS/TCP client open request, using the same AFI, to the source IP address of the HELLO to establish an L3ND TLS/TCP session.

All L3ND PDUs other than HELLO are sent via TLS/TCP, as the server’s destination IP address is known after the HELLO.

When an interface is turned up on a device, it SHOULD issue a HELLO if it is configured to participate in L3ND sessions and repeat HELLOs at a configured interval, with a default of 60 seconds.

If the configured multicast destination address is one that is propagated by switches, the HELLO SHOULD be repeated at a configured interval, with a default of 60 seconds. This allows discovery by new devices which come up on the mesh. In this multi-link scenario, the operator should be aware of the trade-off between timer tuning and network noise and adjust the inter-HELLO timer accordingly.

By default, GTSM, [RFC5082], SHOULD be enabled to test that a received HELLO MUST be on the local link; thus leaving no middle on which a monkey in the middle might stand. It MAY be disabled by configuration. GTSM check failures SHOULD be logged, though with rate limiting to keep from overwhelming logs.

If more than one device responds, one adjacency is formed for each unique source IP address. L3ND treats each adjacency as a separate logical link.

To ameliorate possible load spikes during bootstrap or event recovery, there SHOULD be a jittered delay between receipt of a HELLO and TLS/TCP open. The default delay range SHOULD be zero to five seconds, and MUST be configurable.

If a HELLO is received from an IP Address with which there is an established session for that AFI, the HELLO SHOULD be dropped.

A device with a TLS/TCP listener SHOULD log or otherwise report repeated failed inbound attempts.

6.1. Transport

The Transport signals the type of transport security for the session.

The actual transport options are actually pre-configured in the devices by provisioning, as most require certificates etc. It is best to think of this field as in-band signaling to conform the
correctness of the pre-configurations. Any disagreements MUST BE considered to indicate an error condition and brought to the attention of the operator.

The Transport field is an enumeration with the following values:

0: Raw TCP: TLS is not used.
1: TLS TOFU: TLS using a self-signed server certificate.
2: TLS CA-NoIP: TLS using a CA-Based server certificate, with no IP address extension.
3: TLS CA WithIP: TLS using a CA-Based server certificate, with the server’s IP address in the subject alternative name extension (see [RFC5280] Section 4.2.1.6).
4-255: Reserved.

If server certificates are to be used, they may be locally generated and then signed by a CA or generated by the CA and loaded. See [RFC8635].

6.2. Flags

Though the Working Group scope for this protocol is within a data center, an issue was raised that, on an internet exchange with route server(s), it would attempt to form adjacencies with all members of the exchange. Hence a Flag field is provided to indicate that a device does not intend to field a TLS/TCP server on the announcing interface, but does seek one or more from peers.

Currently, only one Flags field is defined

Bit 0: Client Only This interface does not provide a TLS/TCP server.

Bits 1-7: Reserved.

6.3. Port

The Port is the two octet TCP Port Number (default is TBD3) on which the HELLO sender SHOULD have a waiting TLS/TCP (as specified in Flags) server listening unless the Client Only Flag is set. Though the IANA assigned well-known port SHOULD be used, this field allows configuration of alternate ports.
7. TCP Set-Up

As it is assumed that the configured deployment of a data center would have compatible parameters on all devices, any disagreement over TLS/TCP or trust anchors MUST be logged; with rate limiting of the logging.

By default, GTSM, [RFC5082], SHOULD be enabled to ensure that a SYN received in response to a HELLO is on the local link. It MAY be disabled by configuration. GTSM check failures SHOULD be logged; though with rate limiting to keep from overwhelming logs.

If the receiver of a HELLO agrees with the sender’s choice of TLS/TCP and authentication, both sides have agreed on an AFI for the transport and on each other’s IP address in that AFI. This is sufficient to open a TCP session between them, which will allow for reliable transport of very large data PDUs while obviating the need to invent complex transports.

The L3ND peer with the higher IP address MUST act as the TLS/TCP client and open the transport session (send SYN) toward the peer with the lower IP address.

The server, the sender of the HELLO from the lower IP address, listens on the advertised port for the TLS/TCP session open. The receiver of the acceptable HELLO, the TLS/TCP client, initiates a TLS or raw TCP session toward the sender of the HELLO, the TLS/TCP server, preferably TLS, as advertised. If TLS, the server has chosen and signaled either a self-signed certificate or one configured from the operational CA trusted by both parties, as negotiated in the HELLO exchange.

Once the TLS/TCP session is established, if its interface is configured as point to point, the client side SHOULD stop listening on any port for which it has sent a HELLO. The server, if configured as a point to point interface SHOULD stop sending HELLOS.

If the TLS/TCP open fails, then this SHOULD be logged and the parties MUST go back to the initial state and try HELLO. Logging SHOULD be rate limited.

Should an interface with an established TLS/TCP session be reconfigured changing the TLS/TCP parameters, the TLS/TCP session should be closed or torn down and both parties should return to the HELLO state.
Should the TLS/TCP session terminate for any reason, the devices SHOULD restart/resume HELLOS. When the new TLS/TCP session is started, if possible the OPEN PDU SHOULD try to resume the lost logical session by using the same nonce and resuming from the last Serial Number.

Once the TLS/TCP session has been established, the two devices exchange L3ND PDUs, starting with OPENs.

8. OPEN

Each device has learned the other’s IP Address from the HELLO exchange, see Section 6 and established a TLS/TCP session over a particular AFI.

The first PDU each sends MUST be an OPEN, and the other side MUST respond with an ACK PDU.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Version = 0 |  PDU Type = 1 |             Payload Length
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                      |              Session ID
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                      |              Serial Number
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                  |              AttrCount
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                           Attribute List ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The four octet Payload Length is the number of octets in all fields of the PDU from the Session ID through the Serial Number.

The four octet Session ID is a nonce which uniquely identifies a session. It enables detection of a duplicate OPEN PDU. It SHOULD be either a random number or a high resolution timestamp. It is needed to prevent session closure due to a repeated OPEN caused by a race or a dropped or delayed ACK. It can be used to resume a dropped logical session.

The one octet AttrCount is the number of attributes in the Attribute List. A node may send zero or more attributes.

Attributes are single octets the semantics of which are operator-defined, e.g.: spine, leaf, backbone, route reflector, arabica, ...
Attribute syntax and semantics are local to an operator or datacenter; hence there is no global registry. Nodes exchange their attributes only in the OPEN PDU.

Unlike L3DL [I-D.ietf-lsvr-l3dl], there are no verifiable keys in the PDUs. If the operator wants authentication, integrity, confidentiality, then TLS MUST have been requested by the HELLO and agreed by the TLS session open.

The Serial Number is a monotonically increasing four octet value representing the sender’s state at the time of sending the last PDU. It may be a non-negative integer, a timestamp, etc. If incrementing the Serial Number would cause it to be zero, it should be incremented again.

On session restart (new OPEN, same Session ID), a receiver MAY send the last received Serial Number to tell the sender to only send data with a Serial Number greater (in the [RFC1982] sense), or send a Serial Number of zero to request all data.

This allows a sender of an OPEN to tell the receiver that the sender would like to resume a logical session and that the receiver of the OPEN PDU only needs to send data starting with the PDU with the lowest Serial Number greater (in the [RFC1982] sense) than the one sent in the OPEN. If the sender is not trying to resume a dropped session, the Serial Number MUST be zero.

If the receiver of an OPEN PDU with a non-zero Serial Number can not resume from the requested point, it should return an ACK with an Error Code of 5, Session May Not Be Continued, EType of 1. The sender of the failing OPEN PDU SHOULD respond with an OPEN PDU with a Serial Number of zero.

If a sender of OPEN does not receive an ACK of the OPEN PDU in a configurable time (default 5 seconds), then they SHOULD close or otherwise terminate the TLS/TCP session and restart from the HELLO state.

If an OPEN arrives at L3ND speaker A from B with which A believes it already has an L3ND session (i.e. OPENs have already been exchanged), and the Serial Number in B’s OPEN PDU is non-zero, speaker A SHOULD establish a new sending session by sending an OPEN with the Serial Number being the same as that of A’s last sent and ACKed PDU. A MUST resume sending encapsulations etc. subsequent to the requested Sequence Number. And B MUST retain all previously discovered encapsulation and other data received from A.
If an OPEN arrives at L3ND speaker A from B with which A believes it already has an L3ND session (i.e., OPENs have already been exchanged), and the Serial Number in B’s OPEN is zero, then the A MUST assume that B’s internal state has been reset. All previously discovered encapsulation data MUST BE discarded; and A MUST respond with a new OPEN PDU with a Serial Number of zero.

TCP KeepAlives should be configured and tuned to meet local operational needs. Some defaults and recommendations are needed here.

9. ACK

The ACK PDU acknowledges receipt of a PDU and reports any error condition which might have been raised.

```
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Version = 0  |  PDU Type = 3 |       Payload Length = 6      |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |   ACKed PDU   |     EType     |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |           Error Code          |           Error Hint          |
+---+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The ACK PDU acknowledges receipt of an OPEN, Encapsulation, Vendor PDU, etc. and is used to return error codes if any.

The one octet ACKed PDU is the PDU Type of the PDU being acknowledged, e.g., OPEN, one of the Encapsulations, etc.

If there was an error processing the received PDU, then the one octet EType is non-zero. If the EType is zero, Error Code and Error Hint MUST also be zero.

A non-zero EType is the receiver’s way of telling the PDU’s sender that the receiver had problems processing the PDU. The Error Code and Error Hint will tell the sender more detail about the error.

The decimal value of EType gives a strong hint how the receiver sending the ACK believes things should proceed:

- 0 - No Error, Error Code and Error Hint MUST be zero
- 1 - Warning, something not too serious happened, continue
- 2 - Session should not be continued, try to restart
3 - Restart is hopeless, call the operator

4-15 - Reserved

The two octet Error Code, noting protocol failures, are listed in Section 16.5. Someone stuck in the 1990s might think the catenation of EType and Error Code as an echo of 0x1zzz, 0x2zzz, etc. They might be right; or not.

The two octet Error Hint, is arbitrary additional data the sender of the error PDU thinks will help the recipient or the debugger with the particular error.

9.1. Retransmission

If a PDU sender expects an ACK, e.g. for an OPEN, an Encapsulation, a Vendor PDU, etc., and does not receive the ACK for a configurable time (default five seconds) the TLS/TCP session should be closed or dropped, and both sides revert to HELLO state.

10. The Encapsulations

Once the devices know each other’s IP Addresses, and have established a TLS/TCP session and have successfully exchanged OPENs, the L3ND session is considered established, and the devices SHOULD exchange Layer-3 interface encapsulations, Layer-3 addresses, and Layer-2.5 labels.

Encapsulation data for any AFI/SAFI may be exchanged over a TLS/TCP session irrespective of the AFI/SAFI of the session transport.

The Encapsulation types the peers exchange may be IPv4 (Section 10.3), IPv6 (Section 10.4), MPLS IPv4 (Section 10.6), MPLS IPv6 (Section 10.7), and/or possibly others not defined here.

The sender of an Encapsulation PDU MUST NOT assume that the receiver is capable of the same Encapsulation Type. An ACK (Section 9) with EType of 0 merely acknowledges receipt. Only if both peers have sent the same Encapsulation Type is it safe for Layer-3 protocols to assume that they are compatible for that Encapsulation Type.

A receiver of an encapsulation might recognize an addressing conflict, such as both ends of the link trying to use the same address. In this case, the receiver MUST respond with an error (Error Code 2, Logical Link Addressing Conflict) ACK. As there may be other usable addresses or encapsulations, this error might log and continue, letting an upper layer topology builder deal with what works.
Further, to consider a logical link of a Encapsulation Type to formally be established so that it may be used by other protocols, the addressing for the type must be compatible, e.g. on the same IP subnet.

10.1. The Encapsulation PDU Skeleton

The header for all encapsulation PDUs 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Version = 0  |    PDU Type   |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       |             Count             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        |                 Serial Number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Encapsulation List... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

An Encapsulation PDU describes zero or more addresses of the encapsulation type.

The three octet Count is the number of Encapsulations in the Encapsulation List.

The Serial Number is a monotonically increasing four octet value representing the sender’s state in time. It may be an integer, a timestamp, etc. On session restart (new OPEN), a receiver MAY send the last received Serial Number to request the sender to only send newer data.

If a sender has multiple links on the same interface, separate state: data, ACKs, etc. must be kept for each peer session.

Over time, multiple Encapsulation PDUs may be sent for an interface in a session as configuration changes.

The Receiver MUST acknowledge the Encapsulation PDU with an ACK PDU (Section 9) with the Type field being that of the Type of the Encapsulation PDU being announced, see Section 9.
If the Sender does not receive an ACK in a configurable interval (default five seconds), they SHOULD retransmit. After a user configurable number of failures (default three), the L3ND session should be considered dead, TLS/TCP torn down, and the HELLO process SHOULD be restarted.

If the link is broken below layer-3, retransmission MAY BE retried if data have not changed in the interim and the TLS/TCP session is still alive.

Should an Encapsulation in the Encapsulation List be syntactically invalid, e.g. an out of bounds prefix length, the entire Encapsulation PDU MUST be dropped and the sending party notified by an ACK PDU with an EType of 1 and an Error Code of 3, Encapsulation Error.

10.2. Encapsulation Flags

The one octet Encapsulation Flags field is a sequence of one bit fields as follows:

```
+------------+------------+------------+------------+------------+
|  Ann/With  |   Primary  | Under/Over |  Loopback  | Reserved ..|
+------------+------------+------------+------------+------------+
```

Each encapsulation in an Encapsulation PDU of Type T may announce new and/or withdraw old encapsulations of Type T. It indicates this with the Ann/With Encapsulation Flag, Announce == 1, Withdraw == 0.

Announcing an encapsulation which already exists SHOULD raise an Announce/Withdraw Error (see Section 16.5); the EType SHOULD be 2, suggesting a session restart (see Section 9) so all encapsulations will be resent.

If an interface on a link has multiple addresses for an encapsulation type, one and only one address MAY be marked as primary (Primary Flag == 1) for that Encapsulation Type.

An Encapsulation interface address in an Encapsulation PDU MAY be marked as a loopback, in which case the Loopback bit is set. Loopback addresses are generally not seen directly on an external interface. One or more loopback addresses MAY be exposed by configuration on one or more L3ND speaking external interfaces, e.g. for iBGP peering. They SHOULD be marked as such, Loopback Flag == 1.
Each Encapsulation interface address in an Encapsulation PDU is that of the direct 'underlay interface (Under/Over == 1), or an 'overlay' address (Under/Over == 0), likely that of a VM or container guest bridged or configured on to the interface already having an underlay address.

10.3. IPv4 Encapsulation

The IPv4 Encapsulation describes a device’s ability to exchange IPv4 packets on one or more subnets. It does so by stating the interface’s addresses and the corresponding prefix lengths.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Version = 0  |  PDU Type = 4 |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Count             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Serial Number                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Encaps Flags |          IPv4 Address         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   PrefixLen   |    more ...   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The three octet Count is the sum of the number of IPv4 Encapsulations being announced and/or withdrawn.

10.4. IPv6 Encapsulation

The IPv6 Encapsulation describes a link’s ability to exchange IPv6 packets on one or more subnets. It does so by stating the interface’s addresses and the corresponding prefix lengths.
The three octet Count is the sum of the number of IPv6 Encapsulations being announced and/or withdrawn.

10.5. MPLS Label List

As an MPLS enabled interface may have a label stack, see [RFC3032], a variable length list of labels is needed. These are the labels the sender will accept for the prefix to which the list is attached.

A one octet Label Count of zero is an implicit withdraw of all labels for that prefix on that interface.

The bottom of the stack flag, S, MUST be set on one and only one label. Should this not be the case, the receiver of the erroneous PDU MUST respond with an ACK PDU of EType 1 and Error Code 1, MPLS Error.
10.6. MPLS IPv4 Encapsulation

The MPLS IPv4 Encapsulation describes a logical link’s ability to exchange labeled IPv4 packets on one or more subnets. It does so by stating the interface’s addresses, the corresponding prefix lengths, and the corresponding labels which will be accepted for each address.

```
+--------------------------------------------------+
| Version = 0 | PDU Type = 6 | Payload Length |
+--------------------------------------------------+
| Count      |            |
+--------------------------------------------------+
| Serial Number                                    |
+--------------------------------------------------+
| Encaps Flags | MPLS Label List ... |
+--------------------------------------------------+
| IPv4 Address                                      |
+--------------------------------------------------+
| PrefixLen  | more ...   |
+--------------------------------------------------+
```

The three octet Count is the sum of the number of MPLSv4 Encapsulation being announced and/or withdrawn.

10.7. MPLS IPv6 Encapsulation

The MPLS IPv6 Encapsulation describes a logical link’s ability to exchange labeled IPv6 packets on one or more subnets. It does so by stating the interface’s addresses, the corresponding prefix lengths, and the corresponding labels which will be accepted for each address.
The three octet Count is the sum of the number of MPLSv6 Encapsulations being announced and/or withdrawn.

11. VENDOR - Vendor Extensions

Vendors or enterprises may define TLVs beyond the scope of L3ND standards. This is done using a Private Enterprise Number [IANA-PEN] followed by Enterprise Data in a format defined for that three octet Enterprise Number and one octet Ent Type.

Ent Type allows a Vendor PDU to be sub-typed in the event that the vendor/enterprise needs multiple PDU types.
As with Encapsulation PDUs, a receiver of a Vendor PDU MUST respond with an ACK PDU, possibly signalling an error. Similarly, a Vendor PDU MUST only be sent over an open session.

12. Discussion

This section explores some trade-offs taken and some considerations.

12.1. HELLO Discussion

A device may send IP packets over a Layer-3 interface which transmits data over a single Layer-2 interface or multiple Layer-2 interfaces. Packets sourced by one Layer-3 IP interface over multiple Layer-2 should consider that a Layer-3 interface with multiple Layer-2 interfaces could have many devices which might come at various times, therefore the configured HELLO PDU retransmit time SHOULD be set to a non-zero value, and periodic HELLOs should continue. Packets transmitted on a single Layer-2 interface on a point-to-point (p2p) connection, MAY set the configuration value to zero, so when a TLS/TCP session is up, HELLOs are no longer desirable.

A device with multiple Layer-2 interfaces, traditionally called a switch, may be used to forward packets from multiple devices to one Layer-3 interface, I, on an L3ND speaking device. Interface I could discover a peer J across the switch. Later, a prospective peer K could come up across the switch. If I was not still sending and listening for HELLOs, the potential peering with K could not be discovered. Therefore, on multi-link interfaces, L3ND MUST continue to send HELLOs as long as they are turned up.

13. VLANs/SVIs/Sub-interfaces

One can think of the protocol as an instance (i.e. state machine) which runs on each logical link of a device.

As the upper routing layer must view VLAN topologies as separate graphs, L3ND treats VLAN ports as separate links.

As Sub-Interfaces each have their own layer-3 identities, they act as separate interfaces, forming their own links.

14. Implementation Considerations

An implementation SHOULD provide the ability to configure each logical interface as L3ND speaking or not.
An implementation SHOULD provide the ability to distribute one or more loopback addresses or interfaces into L3ND on an external L3ND speaking interface.

An implementation SHOULD provide the ability to distribute one or more overlay and/or underlay addresses or interfaces into L3ND on an external L3ND speaking interface.

An implementation SHOULD provide the ability to configure one of the addresses of an encapsulation as primary on an L3ND speaking interface. If there is only one address for a particular encapsulation, the implementation MAY mark it as primary by default.

An implementation MAY allow optional configuration which updates the local forwarding table with overlay and underlay data both learned from L3ND peers and configured locally.

15. Security Considerations

For TLS, versions greater than 1.1 MUST be used.

The protocol as is MUST NOT be used outside a datacenter or similarly closed environment without using TLS encapsulation which is based on a configured CA trust anchor.

Many datacenter operators have a strange belief that physical walls and firewalls provide sufficient security. This is not credible. All DC protocols need to be examined for exposure and attack surface. In the case of L3ND, authentication and integrity as provided by TLS validated to a configured shared CA trust anchor is strongly RECOMMENDED.

It is generally unwise to assume that on the wire Layer-3 is secure. Strange/unauthorized devices may plug into a port. Mis-wiring is very common in datacenter installations. A poisoned laptop might be plugged into a device's port, form malicious sessions, etc. to divert, intercept, or drop traffic.

Similarly, malicious nodes/devices could misannounce addressing.

If OPEN PDUs are not over validated TLS, an attacker could forge an OPEN for an existing session and cause the session to be reset.

16. IANA Considerations
16.1. Link Local Layer-3 Addresses

IANA is requested to assign one address (TBD1) for L3DL-L3-LL from the IPv4 Multicast Address Space Registry from the Local Network Control Block (224.0.0.0 - 224.0.0.255 (224.0.0/24)).

IANA is requested to assign one address (TBD2) for L3DL-L3-LL from the IPv6 Multicast Address Space Registry in the the IPv6 Link-Local Scope Multicast address (TBD:2).

16.2. Ports for TLS/TCP

This document requests the IANA to assign a well-known TCP Port Number (TBD3) to the Layer-3 Neighbor Discovery Protocol for the following, see Section 7:

l3nd-server

16.3. PDU Types

This document requests the IANA create a registry for L3ND PDU Type, which may range from 0 to 255. The name of the registry should be L3ND-PDU-Type. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

<table>
<thead>
<tr>
<th>PDU Code</th>
<th>PDU Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>HELLO</td>
</tr>
<tr>
<td>1</td>
<td>reserved</td>
</tr>
<tr>
<td>2</td>
<td>OPEN</td>
</tr>
<tr>
<td>3</td>
<td>ACK</td>
</tr>
<tr>
<td>4</td>
<td>IPv4 Announcement</td>
</tr>
<tr>
<td>5</td>
<td>IPv6 Announcement</td>
</tr>
<tr>
<td>6</td>
<td>MPLS IPv4 Announcement</td>
</tr>
<tr>
<td>7</td>
<td>MPLS IPv6 Announcement</td>
</tr>
<tr>
<td>8-254</td>
<td>Reserved</td>
</tr>
<tr>
<td>255</td>
<td>Vendor</td>
</tr>
</tbody>
</table>

16.4. Flag Bits

This document requests the IANA create a registry for L3ND PL Flag Bits, which may range from 0 to 7. The name of the registry should be L3ND-PL-Flag-Bits. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:
<table>
<thead>
<tr>
<th>Bit</th>
<th>Bit Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Announce/Withdraw (ann == 0)</td>
</tr>
<tr>
<td>1</td>
<td>Primary</td>
</tr>
<tr>
<td>2</td>
<td>Underlay/Overlay (under == 0)</td>
</tr>
<tr>
<td>3</td>
<td>Loopback</td>
</tr>
<tr>
<td>4-7</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

16.5. Error Codes

This document requests the IANA create a registry for L3ND Error Codes, a 16 bit integer. The name of the registry should be L3ND-Error-Codes. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Error Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No Error</td>
</tr>
<tr>
<td>1</td>
<td>MPLS Error</td>
</tr>
<tr>
<td>2</td>
<td>Logical Link Addressing Conflict</td>
</tr>
<tr>
<td>3</td>
<td>Encapsulation Error</td>
</tr>
<tr>
<td>4</td>
<td>Announce/Withdraw Error</td>
</tr>
<tr>
<td>5</td>
<td>Session May Not Be Continued</td>
</tr>
</tbody>
</table>

17. Acknowledgments

The authors thank Ben Maddison and Jeff Haas.

18. References

18.1. Normative References

[I-D.ietf-lsvr-l3dl]


18.2. Informative References

[I-D.ymbk-idr-l3nd-ulpc]


Authors’ Addresses

Randy Bush
Arrcus & Internet Initiative Japan
5147 Crystal Springs
Bainbridge Island, WA 98110
United States of America
Email: randy@psg.com

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America
Email: housley@vigilsec.com

Rob Austein
Arrcus, Inc
Email: sra@hactrn.net
Abstract

This document adds PDUs to the Layer-3 Neighbor Discovery protocol to communicate the parameters needed to exchange inter-device Upper Layer Protocol Configuration for upper-layer protocols such as the BGP family.

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 6 October 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ........................................... 2
2. Reading and Terminology ......................... 2
   3.1. ULPC BGP Attribute sub-TLVs ................. 4
      3.1.1. BGP ASN .................................... 5
      3.1.2. BGP IPv4 Address ......................... 5
      3.1.3. BGP IPv6 Address ......................... 6
      3.1.4. BGP Authentication sub-TLV ............... 6
      3.1.5. BGP Miscellaneous Flags ................. 6
4. Security Considerations ............................. 7
5. IANA Considerations ................................ 7
6. Acknowledgments ..................................... 8
7. References ............................................ 8
   7.1. Normative References ......................... 8
   7.2. Informative References ...................... 9
Authors’ Addresses .................................. 9

1. Introduction

Massive Data Centers (MDCs) which use upper-layer protocols such as BGP4 and other routing protocols may use the Layer-3 Neighbor Discovery Protocol, L3ND, [I-D.ymbk-idr-l3nd] to reveal the inter-device links of the topology. It is desirable for devices to facilitate the configuration parameters of those upper layer protocols to enable more hands-free configuration. This document defines a new L3ND PDU to communicate these Upper-Layer Protocol Configuration parameters.

2. Reading and Terminology

The reader is assumed to have read Layer-3 Neighbor Discovery [I-D.ymbk-idr-l3nd]. The terminology and PDUs there are assumed here.

Familiarity with the BGP4 Protocol [RFC4271] is assumed.
3. Upper-Layer Protocol Configuration PDU

To communicate parameters required to configure peering and operation of Upper-Layer Protocols at IP layer-3 and above, e.g., BGP sessions on a link, a neutral sub-TLV based Upper-Layer Protocol PDU is defined 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 = 0  |    Type = 8   |         Payload Length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |   ULPC Type   |   AttrCount   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Serial Number                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Attribute List ...                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Version, Type, and Payload Length as defined in [I-D.ymbk-idr-l3nd] apply to this PDU.

The BGP Authentication sub-TLV provides for provisioning MD5, which is a quite weak hash, horribly out of fashion, and kills puppies. But, like it or not, it has been sufficient against the kinds of attacks BGP TCP sessions have endured. So it is what BGP deployments use.

As the ULPC PDU may contain keying material, e.g. [RFC2385], it should be over TLS.

ULPC Type: A one byte integer denoting the type of the upper-layer protocol

0 :   Reserved
1 :   BGP
2-255 :  Reserved

The one octet AttrCount is the number of attribute sub-TLVs in the Attribute List.

The Attribute List is a, possibly null, set of sub-TLVs describing the configuration attributes of the specific upper-layer protocol.

An Attribute consists of a one octet Attribute Type, a one octet Attribute Length of the number of octets in the Attribute, and a Payload of arbitrary length up to 253 octets.
3.1. ULPC BGP Attribute sub-TLVs

The parameters needed for BGP peering on a link are exchanged in sub-TLVs within an Upper-Layer Protocol PDU. The following describe the various sub-TLVs for BGP.

The goal is to provide the minimal set of configuration parameters needed by BGP OPEN to successfully start a BGP peering. The goal is specifically not to replace or conflict with data exchanged during BGP OPEN. Multiple sources of truth are a recipe for complexity and hence pain.

If there are multiple BGP sessions on a link, e.g., IPv4 and IPv6, then separate BGP ULPC PDUs should be sent, one for each address family.

A peer receiving BGP ULPC PDUs has only one active BGP ULPC PDU for an particular address family on a specific link at any point in time; receipt of a new BGP ULPC PDU for a particular address family replaces the data any previous one; but does not actually affect the session.

If there are one or more open BGP sessions, receipt of a new BGP ULPC PDU SHOULD not affect these sessions. The received data are stored for a future session restart.

As a link may have multiple encapsulations and multiple addresses for an IP encapsulation, which address of which encapsulation is to be used for the BGP session MUST be specified.

For each BGP peering on a link here MUST be one agreed encapsulation, and the addresses used MUST be in the corresponding L3ND IPv4/IPv6 Encapsulation PDUs. If the choice is ambiguous, an Attribute may be used to signal preferences.

If a peering address has been announced as a loopback, i.e. MUST BE flagged as such in the L3ND Encapsulation PDU (see [I-D.ymbk-idr-l3nd] Sec. 10.2), a two or three hop BGP session MUST be established as needed. Otherwise a direct one hop session is used. The BGP session to a loopback will forward to the peer’s address which was marked as Primary in the L3ND Encapsulation Flags, iff it is in a subnet which is shared with both BGP speakers. If the
primary is not in a common subnet, then the BGP speaker MAY pick a forwarding next hop that is in a subnet they share. If there are multiple choices, the BGP speaker SHOULD have signaled which subnet to choose in an Upper-Layer Protocol Configuration PDU Attribute.

Attributes MUST be unique in the Attribute List. I.e. a particular Attr Type MUST NOT occur more than once in the Attribute List. If a ULPC PDU is received with more than one occurrence of a particular Attr Type, an Error ACK MUST be returned.

As there are separate PDU Attr Types for IPv4 and IPv6 peering addresses, separate sessions for the two AFIs MAY be created for the same ASN in one ULPC PDU.

3.1.1. BGP ASN

The four octet Autonomous System number MUST be specified. If the AS Number is less than 32 bits, it is padded with high order zeros.

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Attr Type = 1 | Attr Len = 4 |             My ASN            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-

3.1.2. BGP IPv4 Address

The BGP IPv4 Address sub-TLV announces the sender’s four octet IPv4 BGP peering source address and one octet Prefix Lenth to be used by the receiver. At least one of IPv4 or IPv6 BGP source addresses MUST be announced.

As usual, the BGP OPEN capability negotiation will determine the AFI/SAFIs to be transported over the peering, see [RFC4760].

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Attr Type = 2 | Attr Len = 5 | My IPv4 Peering Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Prefix Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
3.1.3. BGP IPv6 Address

The BGP IPv6 Address sub-TLV announces the sender’s 16 octet IPv6 BGP peering source address and one octet Prefix Length to be used by the receiver. At least one of IPv4 or IPv6 BGP source addresses MUST be announced.

As usual, the BGP OPEN capability negotiation will determine the AFI/SAFIs to be transported over the peering, see [RFC4760].

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Attr Type = 3 | Attr Len = 17 |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| My IPv6 Peering Address                                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Prefix Len |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

3.1.4. BGP Authentication sub-TLV

The BGP Authentication sub-TLV provides any authentication data needed to OPEN the BGP session. Depending on operator configuration of the environment, it might be a simple MD5 key (see [RFC2385]), the name of a key chain in a KARP database (see [RFC7210]), or one of multiple Authentication sub-TLVs to support hop[RFC4808].

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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Attr Type = 4 |    Attr Len   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                   BGP Authentication Data ... |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

3.1.5. BGP Miscellaneous Flags

The BGP session OPEN has extensive, and a bit complex, capability negotiation facilities. In case one or more extra attributes might be needed, the two octet BGP Miscellaneous Flags sub-TLV may be used.
4. Security Considerations

All the Security considerations of [I-D.ymbk-idr-l3nd] apply to this PDU.

As the ULPC PDU may contain keying material, see Section 3.1.4, it SHOULD BE over TLS, not clear TCP.

Any keying material in the PDU SHOULD BE salted and hashed.

The BGP Authentication sub-TLV provides for provisioning MD5, which is a quite weak hash, horribly out of fashion, and kills puppies. But, like it or not, it has been sufficient against the kinds of attacks BGP TCP sessions have endured. So it is what BGP deployments use.

5. IANA Considerations

This document requests the IANA create a new entry in the L3ND PDU Type registry as follows:

<table>
<thead>
<tr>
<th>PDU Code</th>
<th>PDU Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>ULPC</td>
</tr>
</tbody>
</table>
This document requests the IANA create a registry for L3ND ULPC Type, which may range from 0 to 255. The name of the registry should be L3ND-ULPC-Type. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>BGP</td>
</tr>
<tr>
<td>2-255</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

6. Acknowledgments

The authors thank Rob Austein, Sue Hares, and Russ Housley.

7. References

7.1. Normative References


[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an
IANA Considerations Section in RFCs", RFC 5226,
DOI 10.17487/RFC5226, May 2008,

(BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010,

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

7.2. Informative References

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5
Signature Option", RFC 2385, DOI 10.17487/RFC2385, August

[RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5",
RFC 4808, DOI 10.17487/RFC4808, March 2007,

[RFC7210] Housley, R., Polk, T., Hartman, S., and D. Zhang,
"Database of Long-Lived Symmetric Cryptographic Keys",
RFC 7210, DOI 10.17487/RFC7210, April 2014,

Authors’ Addresses

Randy Bush
Arrcus & IIJ
5147 Crystal Springs
Bainbridge Island, WA 98110
United States of America
Email: randy@psg.com

Keyur Patel
Arrcus
2077 Gateway Place, Suite #400
San Jose, CA 95119
United States of America
Email: keyur@arrcus.com
BGP SR Policy Extensions for metric
draft-zhang-idr-sr-policy-metric-00

Abstract

SR Policy candidate paths can be represented in BGP UPDATE messages. BGP can then be used to propagate the SR Policy candidate paths to the headend nodes in the network. After SR Policy is installed on the ingress node, the packets can be steered into SR Policy through route selection. Therefore, route selection may be performed on the ingress node of the SR Policy. If there are multiple routes to the same destination, the route selection node can select routes based on the local policy. The local policy may use the IGP metric of the selected path, which is the IGP Metric of the SR Policy. Thus the BGP UPDATE message need carry the metric of each segment list of the SR Policy Candidate Path, which can be used in path selection of routing.

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 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 6 September 2022.
1. Introduction

[I-D.ietf-idr-segment-routing-te-policy] defines SR Policy and Tunnel Encapsulation Attributes. It defines the segment list of the SR policies. Each segment list of an SR Policy is an segment routing path, which may be calculated by path compuation element and delivered to the head node of the device by BGP Update Message. On the ingress node, when steer traffic to an SR Policy, the ingress node may need to select between multiple SR Policy paths. And the selection policy may need the path metric information. Therefore, BGP needs to carry the metric of each path when delivering the segment list of the SR Policy through Update messages to facilitate route selection on the device.

2. Terminology

The following terminology is used in this document.

SR Policy: An ordered list of segments.


3. Motivation

In route selection scenarios, the metric of the SR Policy segment list may be required.

The specific scenarios are as follows:

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

|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |     |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

On PE1, the route prefix to CE1 has two different next hop, PE2 and PE3. The next hop to PE1 uses an SR Policy1 on PE1, the endpoint of SR Policy1 is PE2. The next hop to PE2 uses an SR Policy2 on PE1, the endpoint of SR Policy2 is PE3. The prefix to CE1 want to choose a next hop based on the IGP metric of the route PE1 to PE2 and PE1 and PE3, which uses SR Policy1 and SR Policy2. Thus need the IGP metric of SR Policy segment list on PE1.

4. SR Policy and Tunnel Encapsulation Attribute Update

As the metric is defined, the tunnel attribute encapsulation of the BGP SR Policy needs to be updated.

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

Policy Candidate Path Name

Explicit NULL Label Policy (ENLP)

Segment List

Weight

Metric

Segment

Segment

....

....

Where metric indicates the metric for the segment list.

4.1. Metric sub-TLV

A new sub-TLV called Metric sub-TLV is defined. Metric sub-TLV specifies the metric of an SR policy segment list. Each sub-TLV is encoded as shown in Figure 1.
Figure 1: Metric Sub-TLV

Type: Metric, 1 octet, TBD.

Length: 6.

Metric Type: 1 octet. The Type of metric, can be IGP metric, TE metric, delay, etc.

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.

Metric Value: a 4-octet value.

5. Metric process of SR Policy segment list

When SR Policy headend get the SR Policy segment list with metric, how to process the metric is local policy.

The active candidate path of SR Policy may have several segment lists, each segment list have different metric. It is recommended that the segment lists in one candidate path have the same metric type. If the metric value of segment lists in one candidate path is different, the candidate path metric can use the minimum value as the metric of candidate path. And the SR Policy metric use the metric value of active candidate path.

6. Acknowledgements

TBD.

7. IANA Considerations

This document requests that IANA allocates a new sub-TLV type as defined in Section 4.1 from the "Sub-TLVs for SR Policy" registry as specified.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Metric</td>
<td>This document</td>
</tr>
</tbody>
</table>

Figure 2: Template ID sub-TLV

8. Security Considerations

These extensions to BGP SR Policy do not add any new security issues to the existing protocol.
9. References

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

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


Authors’ Addresses

Ka Zhang
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China
Email: zhangka@huawei.com
Jie Dong
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing
100095
China
Email: dongjie@huawei.com
Inter-domain Network Slicing via BGP-LU
draft-zhou-idr-inter-domain-lcu-04

Abstract

This document aims to solve inter-domain network slicing problems using existing technologies. It attempts to establish multiple BGP-LU LSPs of different colors for a/multiple prefix to stitch multiple network segments.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.
1.  Introduction

In the traditional end to end inter-domain network slicing, BGP-LU is used to build inter-domain MPLS LSP, and overlay service will be directly over BGP-LU LSP. For an E2E BGP-LU LSP, if overlay service has TE requirements that defined by a color, the BGP-LU LSP need also have a sense of color, i.e., BGP-LU label could be allocated per color.

[RFC8277] specifies a set of procedures for using BGP to advertise that a specified router has bound a specified MPLS label to a specified address prefix. It’s an effective way for inter-domain labels, but it does not have the ability to select the underlying network resources.

This document describes the colored BGP-LU LSP, which contains two options:

* One is to define the multiple paths for the same destination prefix and advertise in BGP UPDATE message, and each UPDATE message can contain the color Extended Community [RFC9012] with different color value, which helps to select the underlying resources. This mode require additional path function defined in [RFC7911].

* The other is that multiple prefixes and multiple colors are configured on PE. One prefixes corresponds to one color. This mode does not require to additional path function.
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. Color

[RFC9012] introduces the concept of color, which is used as one of the KEY of SR policy [I-D.ietf-spring-segment-routing-policy]. The color of SR policy defines a TE purpose, which includes a set of constraints such as bandwidth, delay, TE metric, etc.

To help routing decisions, each UPDATE may contain a Color Extended Community with a specific color value, the Color Sub-TLV is only an opaque extended community.

3. Advertising multiple paths

A BGP speaker can advertise multiple paths for a particular address prefix by a Path identifier in the Extended NLRI Encoding as defined in [RFC7911].

```
+--------------------------------+
| Path Identifier (4 octets)    |
+--------------------------------+
| Length (1 octet)              |
+--------------------------------+
| Prefix (variable)             |
+--------------------------------+
```

Figure 1

The Path Identifier only identifies a path, not carrying any particular semantics. In this document, it can be generated by the <Prefix,Color> tuple. The assignment to the Path Identifier for a path by a BGP speaker is purely a local matter.

Therefore, if a BGP speaker has two colors for the prefix P, which correspond to two different paths, it may advertise two UPDATE NLRIs, <prefix, pathid1> with color1 extended community and <prefix, pathid2> with color2 extended community. Pathid1 and pathid2 in two UPDATE NLRIs MUST be different.

Note that in this document, BGP speakers acting as border routers that interact with external neighbors need to support advertising multiple paths corresponding to the same prefix. Although multiple
paths have different path IDs, they have the same next hop. As for the procedures of mutual backup paths with the same prefix and the different next hops, refer to [RFC7911].

4. Assigning Label(s)

[RFC8277] describes how to use BGP to bind MPLS label(s) to the address prefixes. The specific format of the UPDATE message is detailed in Section 2 of [RFC8277].

[RFC8277] Section 3.2 details the process of modifying the Label field during propagation. When propagating a SAFI-4 or SAFI-128 route, if the Network Address of Next Hop field has never changed, the label field must remain unchanged. Otherwise, if the Network Address of Next Hop field is changed, the label field(s) of the propagating route must contain the label(s) that is (are) bound to the prefix at the new next hop. What the label changes to depends on the local policy. However, LSPs with different color paths need to have different label(s).

5. Inter-domain Network Slicing via BGP-LU

5.1. Colored BGP-LU Capability Advertisement

A BGP speaker that uses Colored BGP-LU Extensions SHOULD use the Capability Advertisement procedures [RFC3392] to determine whether the speaker could use Colored BGP-LU Extensions with a particular peer.

The fields in the Capabilities Optional Parameter are set as follows:

* The Capability Code field TBD1 (which indicates Colored BGP-LU Extensions capabilities).

* The Capability Length field is set to 4.

* The Capability Value field is defined as:

```
+-----------------------------------------------+
| Address Family Identifier (2 octets)          |
+-----------------------------------------------+
| Subsequent Address Family Identifier (1 octet)|
+-----------------------------------------------+
| reserve (1 octet)                            |
+-----------------------------------------------+
```
Figure 2

where:

AFI-Address Family Identifier (16 bit), The values is 1 "IPV4" or 2 "IPV6".

SAFI-Subsequent Address Family Identifier (8 bit), The values is 1 "Unicast" or 4 (BGP LU).

Res.-Reserved (8 bit) field. SHOULD be set to 0 by the sender and ignored by the receiver.

Note that not setting the field value to 0 may create issues for a receiver not ignoring the field. In addition, this definition is problematic if it is ever attempted to redefine the field.

5.2. Colored BGP-LU realized

[RFC7911] defined that multiple paths for a particular address prefix by a Path identifier can be advertised in BGP UPDATE message, and each UPDATE message can contain the Color Extended Community [RFC9012] with different color value. That is a simple existing way to realize BGP-LU color function, and only an extension of Colored BGP-LU capability advertisement is required.

Consider the following example of establishing multiple BGP-LU LSPs per different colors in a cross-domain scenario.
In figure 1, PE1 advertises two paths: `<1.1.1.1, path-id1>` carries the color1 attribute and `<1.1.1.1, path-id2>` carries the color2 attribute to ASBR1. PE1 advertises the binding between the prefix 1.1.1.1 and label 200. Because of the end node, both paths have the same label value 200.

ASBR1 receives these two paths from PE1, and when sending to ASBR2, it modifies the next hop to itself. And allocate two new labels based on `<prefix, path-id, color>`. As shown in Figure 1, ASBR1 sends two paths to ASBR2, `<1.1.1.1, path-id1>` carries color1+label201, and `<1.1.1.1, path-id2>` carries color2+label202.

Similarly, ASBR2 also generates two different labels based on the `<prefix, path-id, color>`. As shown in Figure 1, multiple end to end BGP-LU LSPs are established. Different BGP-LU LSPs select the underlay SR-BE/TE tunnels according to their colors.
6. SRv6 support

Colored BGP-LU can be also used to setup end-to-end color-aware connectivity using Segment Routing over IPv6 (SRv6) [RFC8402].

As defined in [I-D.ietf-bess-srv6-services], to provide SRv6 service with underlay SRv6 policy connectivity, the egress PE signals the BGP overlay service route with SRv6 Service SID and color extended community. The ingress PE encapsulates the payload in an outer IPv6 header which contains the underlay SRv6 policy segment list and the overlay Service SID.

In addition, another solution is to provide SRv6 service with underlay SRv6 best-effort connectivity that is created by global IPv6 (AFI/SAFI 2/1) with color extended community. The underlay SRv6 SID is allocated based on <global IPv6, path-id, color>. The ingress PE encapsulates the payload in an outer IPv6 header which contains the underlay SRv6 SID and the Service SID.

7. Deploy Considerations

All BGP routers (PE1--ASBR1, ASBR1---ASBR2, ASBR2---PE2) SHOULD be Colored BGP-LU neighbors in advance. There may be multiple border routers to ensure multipath backup. All routers require the Colored BGP-LU Capability Advertisement. If transit network domains that do not support Colored BGP-LU

* When the Colored BGP-LU neighbor receives the BGP-LU routes, if it continues to advertise the BGP-LU routes to the upstream neighbor that supports the Colored BGP-LU, the BGP-LU routes shouldn’t be changed to the Colored BGP-LU routes.

* When receiving the Colored BGP-LU advertisement from the neighbor that supports Colored BGP-LU, if the advertisement continues to be advertised to the upstream neighbor that does not support Colored BGP-LU, the advertisement should be changed to BGP-LU advertisement, that is, advertise one out of multiple path.

This document not only supports interprovider VPNs while the customer sites belong to different ASs, but also supports the Carrier-of-Carriers VPNs while the customer site belong to the same AS. Multiple operators are involved, so AS border routers may involve color mapping, color namespaces, or color service chains. These services can be delivered by the controller configurations or the local configurations.
8. Acknowledgements

TBD.

9. IANA Considerations

TBD.

10. Security Considerations

TBD.

11. Normative References

[I-D.ietf-bess-srv6-services]
Dawra, G., Filisfils, C., Talaualikar, K., Raszuk, R.,
Decraene, B., Zhuang, S., and J. Rabadan, "SRv6 BGP based
Overlay Services", Work in Progress, Internet-Draft,
draft-ietf-bess-srv6-services-12, 5 March 2022,
<https://datatracker.ietf.org/doc/html/draft-ietf-bess-
srv6-services-12>.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaualikar, K., Voyer, D., Bogdanov, A., and
P. Mattes, "Segment Routing Policy Architecture", Work in
Progress, Internet-Draft, draft-ietf-spring-segment-
routing-policy-19, 5 March 2022,
<https://datatracker.ietf.org/doc/html/draft-ietf-spring-
segment-routing-policy-19>.

[RFC2119]  Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC7911]  Walton, D., Retana, A., Chen, E., and J. Scudder,
"Advertisement of Multiple Paths in BGP", RFC 7911,
DOI 10.17487/RFC7911, July 2016,

[RFC8277]  Rosen, E., "Using BGP to Bind MPLS Labels to Address
Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017,

Decraene, B., Litkowski, S., and R. Shakir, "Segment
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,
Authors’ Addresses

Ran Chen
ZTE Corporation
Nanjing
China
Email: chen.ran@zte.com.cn

Chunning Dai
ZTE Corporation
Nanjing
China
Email: dai.chunning@zte.com.cn

Shaofu Peng
ZTE Corporation
Nanjing
China
Email: peng.shaofu@zte.com.cn