

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: January 13, 2022

P. Bergeon, Ed.
Nokia
July 12, 2021

Flowspec TTL (Time to Live) Match
draft-bergeon-flowspec-ttl-match-00

Abstract

This document defines a new component type to match TTL (Time to Live) values using BGP Flow Specification 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 January 13, 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.

Table of Contents

1. Introduction	2
2. Definitions of Terms Used in This Memo	2
3. Motivation	2
4. Specification	3
5. Ordering of Flow Specifications Considerations	3
6. Security Considerations	3
7. IANA Considerations	4
8. References	4
8.1. Normative References	4
8.2. URIs	5
Author's Address	5

1. Introduction

One general purpose of BGP Flowspec [RFC8955] is to distribute firewall rules, also known as filtering or Access Control Lists (ACLs) rules, in receiving routers for mitigation of denial-of-service (DoS) attacks using flows specifications encoded as BGP NLRI [RFC4271].

BGP Flowspec [RFC8955] defines 12 component types that can be used as match criteria in filtering rules with destination prefix, source prefix, IP protocol, port, destination port, source port, ICMP type, ICMP code, TCP flags, packet length, DSCP and fragment.

The IP header field Time to Live (TTL) is a notable absent of the component types defined in BGP Flowspec [RFC8955].

This document proposes to address this by adding support for a new Flowspec component type to add support for TTL match.

2. Definitions of Terms Used in This Memo

NLRI - Network Layer Reachability Information.

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.

3. Motivation

As defined in [RFC0791], the TTL is an indication of an upper bound on the lifetime of an internet datagram. It is set by the sender of

the datagram and reduced by each router along the route where it is processed.

This unique property of the IP header TTL field can make it particularly useful for security and distributed-denial-of-service (DDoS) mitigation.

Studies such as the one presented at NANOG 82 Tracing DDoS End-to-End [1] highlight how filtering traffic based on TTL values can be used as an effective mitigation for DDoS attacks at the IP edge of the network. In particular, the TTL value can be used to differentiate legitimate traffic from DDoS attack traffic generated by DDoS for hire services.

Different IP edge routers of a given network under attack may see the same attack traffic using slightly different TTL values, however these routers can use a common set of filtering rules propagated via BGP flowspec to mitigate the attack using range(s) of TTL values.

4. Specification

This document defines a new Flowspec component type, value TBD, named "TTL (Time to Live)".

Encoding: <type (1 octet), [numeric_op, value]+>

Defines a list of {numeric_op, value} pairs used to match the 8-bit TTL field value.

This component type uses the Numeric Operator (numeric_op) as defined already in [RFC8955] section 4.2.1.1.

Type TBD component values SHOULD be encoded as single octet (numeric_op len=00).

5. Ordering of Flow Specifications Considerations

The ordering of Flow Specifications rules defined in [RFC8955] remains unchanged and applies to the component type introduced in this document.

6. Security Considerations

The new component type introduced in this document does not introduce new security considerations other than the ones already defined in [RFC8955].

7. IANA Considerations

IANA is requested to assign a type from the First Come First Served range of the "Flow Spec Component Types" registry:

Type Value	Name	Reference
TBD	TTL (Time to Live)	this document

Reference: this document

Registry Owner/Change Controller: IESG

Registration procedures:

Range	Registration Procedures
0-127	IETF Review
128-249	First Come First Served
250-254	Experimental
255	Reserved

Note: a separate "owner" column is not provided because the owner of all registrations, once made, is "IESG".

8. References

8.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8955] Loibl, C., Hares, S., Raszuk, R., McPherson, D., and M. Bacher, "Dissemination of Flow Specification Rules", RFC 8955, DOI 10.17487/RFC8955, December 2020, <<https://www.rfc-editor.org/info/rfc8955>>.

8.2. URIs

- [1] <https://www.nanog.org/news-stories/nanog-tv/nanog-82-webcast/tracing-ddos-end-to-end-in-2021/>

Author's Address

Philippe Bergeon (editor)
Nokia

Email: philippe.bergeon@nokia.com

Internet Engineering Task Force
Internet Draft
Updates: 4271 (if approved)
Intended status: Standards Track
Expires: February 13, 2022

E. Chen
J. Yuan
Palo Alto Networks
August 12, 2021

Deterministic Route Redistribution into BGP
draft-chen-bgp-redist-03.txt

Abstract

In this document we present several examples of non-deterministic routing behavior involving route redistribution into BGP. In order to eliminate such non-deterministic behavior, we propose an enhancement to BGP route selection that would take into account the administrative distance under certain conditions. We also recommend that the LOCAL_PREF value be reduced for the redistributed backup route, and be calculated automatically based on the administrative distance.

Status of this Memo

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

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

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

This Internet-Draft will expire on February 13, 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

A routing protocol usually downloads its best (or active) route to the routing table, also known as Routing Information Base (RIB), which in turn selects the best (or active) route to program the forwarding table.

When comparing routes from different routing protocols, RIB typically uses the "administrative distance" [ADMIN-DIS] (abbreviated as "admin-distance" hereafter) as the tie breaker. The convention is that a route with a lower admin-distance is more preferred, and that is assumed in this document when specific admin-distance values are given as examples. The admin-distance associated with a route in RIB is commonly used to implement various routing schemes such as designating primary and backup routes in a network.

On the other hand, the route selection in BGP [RFC4271] involves comparing the LOCAL_PREF, AS_PATH and other BGP attributes. The bestpath in BGP usually becomes the candidate for downloading to the RIB, and for advertising to BGP neighbors.

It is common to redistribute routes from other routing protocols (such as "static routing" [STATIC-R]) into BGP for route propagation. This topic is briefly discussed in [Sect. 9.4, RFC4271]. A redistributed route is usually assigned a fixed LOCAL_PREF value, and has an empty AS_PATH attribute.

The interaction between RIB and BGP follows these general rules:

- o A local route may be redistributed into BGP only if it is active in RIB based on the admin-distance.
- o Only the bestpath in BGP is downloaded to RIB.

Currently the admin-distance does not play any role in BGP route selection. Due to the lack of such correlation between RIB and BGP, when a backup route (based on the admin-distance) is redistributed into BGP as shown in the next section, routing may converge to different paths depending on the order of path arrivals. Such non-deterministic routing behavior is clearly detrimental to network operations.

In order to eliminate such non-deterministic behavior, we propose an enhancement to BGP route selection that would take into account the admin-distance under certain conditions. We also recommend that the LOCAL_PREF value be reduced for the redistributed backup route, and be calculated automatically based on the admin-distance.

The proposed enhancement and recommendation are backward compatible, and can be deployed on an individual router basis.

Although the static routing is used as examples in the document, the proposed enhancement and recommendation also apply when a route is redistributed from other routing protocols into BGP.

1.1. Requirements Language

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

2. The Problem

In this section several examples are presented to illustrate the non-deterministic routing behavior involving route redistribution into BGP.

2.1. On a Single Router

Consider an example in which there are two paths for the same destination on a single router. As shown in the following table, the primary path A is received from an external BGP neighbor, and the backup path B is a static route and is configured for redistribution into BGP.

Path	Type	Admin_Distance	LOCAL_PREF	AS_PATH
A	EBGP	20	100	65535
B	Static	150	100	--

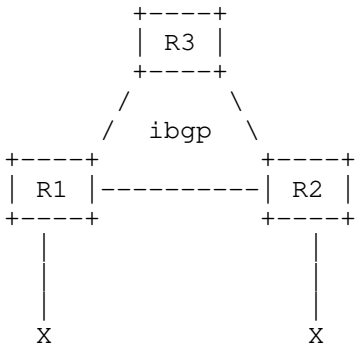
Depending on the order of path arrivals, the path that arrives first would be selected as the bestpath in both RIB and BGP.

More specifically, if Path A is received in BGP and is downloaded to RIB first, it would remain as the best in RIB (due to the admin-distance) even when Path B shows up in RIB later. In this case Path A would be the best one in both RIB and BGP.

If Path B shows up in RIB and is redistributed into BGP first, it would remain as the best in BGP (due to it being a local route or with a shorter AS-PATH) even when Path A is received in BGP later. In this case Path B would be the best one in both RIB and BGP.

2.2. Network-wide Behavior

Consider the following example in which Routers R1, R2 and R3 are part of a provider network and IBGP sessions are maintained among them. There are two customer connections, a primary connection on R1 and a backup connection on R2. The customer route X is statically routed on both R1 and R2, and is redistributed into BGP. On R2, the backup path for X is configured with a less preferred admin-distance than the one for IBGP paths.



While R1 consistently selects the local static route as the best one, the route selection on R2 would be non-deterministic. As shown in the following figure, there are potentially two BGP paths A and B for X on R2, with Path A learned from R1 and Path B locally redistributed.

Path	Type	Admin_Distance	LOCAL_PREF	AS_PATH
A	IBGP	200	100	--
B	Static	210	100	--

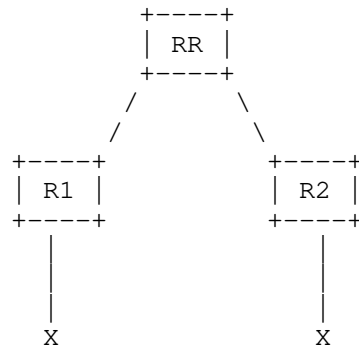
Depending on the order of arrivals of these two paths, the path that arrives first would be selected as the bestpath in both RIB and BGP.

More specifically, if Path A is received in BGP and is downloaded to RIB first, it would remain as the best in RIB (due to the admin-distance) even when Path B shows up in RIB later. In this case A would be the best one in both RIB and BGP.

If Path B shows up in RIB and is redistributed into BGP first, it would remain as the best in BGP (due to it being a local route or with a lower IGP metric) even when Path A is received in BGP later. In this case Path B would be the best one in both RIB and BGP.

The non-deterministic route selection on R2 may cause other nodes (like R3) to converge to different paths as well. The routing behavior in the network would be non-deterministic, and inconsistent with the intended routing design.

A network using BGP route reflection [RFC4456] (or BGP confederation [RFC5065]) may experience additional cases of network-wide "non-deterministic" routing behavior. For example in the following figure, when both R1 and R2 advertise their respective local routes to the route reflector (RR) simultaneously, the RR would use the "IGP metric" to choose the bestpath between the two IBGP paths. As a result the network may or may not converge to the primary path.



3. The Proposed Solution

In order to eliminate the non-deterministic routing behavior involving route redistribution into BGP, we propose an enhancement to BGP route selection that would take into account the admin-distance under certain conditions. We also recommend that the LOCAL_PREF value be reduced for the redistributed backup route, and calculated automatically based on the admin-distance.

3.1. Enhancement to BGP Route Selection

To make it deterministic on a single router regarding the route being sourced and advertised to the network, we propose that the following procedure be added prior to the step that compares the degrees of preference of routes and identifies the route that has the highest degree of preference, as described in Sect. 9.1.2 [RFC4271] for BGP route selection:

When comparing a locally redistributed route with another route that is either locally aggregated or received from an external neighbor, favor the one with a more preferred admin-distance. The admin-distance for a BGP route is obtained as follows:

For a locally redistributed route, it is inherited from the route being redistributed from RIB.

For a non-redistributed route, it is of the same value as the admin-distance assigned to the route for the purpose of RIB installation (regardless of whether it is actually installed in RIB).

It should be noted that IBGP paths are deliberately excluded from the algorithm. As the admin-distance is not propagated by BGP, involving IBGP paths in the admin-distance comparison can easily result in unintended routing behavior and even route churns. To influence route selection in a network, use the LOCAL_PREF attribute as described in the next section.

3.2. Setting the LOCAL_PREF Value

When a non-BGP route is designated as a backup route in the network, it should be assigned a less preferred admin-distance than the value for IBGP routes. When such a route is redistributed into BGP, the LOCAL_PREF value for the redistributed route SHOULD be set lower than the LOCAL_PREF values of the primary route and other more preferred routes.

Assuming the default LOCAL_PREF value is assigned to the primary route, then the LOCAL_PREF value for the redistributed backup route can be calculated automatically as described by the following pseudo-code:

```
if (redist_admin_distance > ibgp_admin_distance) {
    offset = redist_admin_distance - ibgp_admin_distance;
    if (default_local_pref > offset)
        calculated_local_pref = default_local_pref - offset;
    else
        calculated_local_pref = 0;
}
```

in which

- o "redist_admin_distance" is the admin-distance of the route being redistributed.
- o "ibgp_admin_distance" is the admin-distance for IBGP routes on the local router.
- o "default_local_pref" is the default LOCAL_PREF value in the network.
- o "calculated_local_pref" is the calculated LOCAL_PREF value for the redistributed route.

Clearly, in order for the calculated LOCAL_PREF value to truly reflect the intended routing design, the admin-distance needs to be assigned properly. Guideline is provided on assigning the admin-distance in the next section.

This algorithm would not apply if the "default_local_pref" is not assigned to the primary route, in which case manual configuration should be used.

In addition to lowering the LOCAL_PREF value, it may be necessary to modify the parameters for the aforementioned redistributed route pertaining to any vendor-specific route selection criteria preceding the LOCAL_PREF comparison. For example, the "weight" parameter exists in a number of implementations in which case the "weight" for the aforementioned redistributed route should be made equal to the default "weight" for IBGP routes.

3.3. Admin-distance Assignment

In order to achieve the desired routing scheme using the LOCAL_PREF calculated from the admin-distance, coordination would be necessary for the admin-distance assignment when the same destination is redistributed from multiple routers in a network.

While the default LOCAL_PREF value is usually consistent in a network, the default admin-distance for IBGP routes can vary from one node to another in a multi-vendor network.

The coordination of the admin-distance assignment can be simplified by examining the "role" that a non-BGP route is supposed to play (such as being the primary, the secondary or the tertiary), and then associate an "offset" to the route based on its role. Among the routes involved, the less preferred a route is, the higher the offset should be. Then the admin-distance for the route can be assigned as (ibgp_admin_distance + offset), and the desired LOCAL_PREF value would be automatically calculated using the algorithm described in the previous section.

As an example shown in the following table, there are three non-BGP paths for the same destination on separate routers A, B and C in the network and they are designated as the primary, the secondary and the tertiary. The default LOCAL_PREF value is 100 in the network, and the "ibgp_admin_distance" is 200 on the router with the secondary path, and 170 on the router with the tertiary path.

The desired LOCAL_PREF values for the redistributed routes are obtained using the algorithm and procedures described in this document.

Router	Role	Offset	Admin_Distance	LOCAL_PREF
A	Primary	-	50	100 (Default)
B	Secondary	10	200 + 10	100 - 10
C	Tertiary	20	170 + 20	100 - 20

3.4. Configuration Option

Configuration can be used to achieve the equivalent outcome by setting the appropriate LOCAL_PREF value (and also the "weight" parameter if applicable) for the redistributed backup route. It can also be used to override the LOCAL_PREF value calculated based on the admin-distance value of the redistributed route as proposed in this document.

When route redistribution is part of a more complex routing scheme beyond what can be automated with the proposed solution, configuration can also be used following the general principles discussed in this document.

4. IANA Considerations

This document has no request for IANA.

5. Security Considerations

The solution proposed in this document does not change the underlying security or confidentiality issues inherent in the existing BGP [RFC4271].

6. Acknowledgments

The authors would like to thank Naiming Shen, Acee Lindem and Robert Raszuk for inputs and discussions.

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<http://www.rfc-editor.org/info/rfc4271>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

7.2. Informative References

- [STATIC-R] Static routing, Wikipedia,
https://en.wikipedia.org/wiki/Static_routing
- [ADMIN-DIS] Administrative distance, Wikipedia,
https://en.wikipedia.org/wiki/Administrative_distance.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<http://www.rfc-editor.org/info/rfc4456>>.
- [RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <<http://www.rfc-editor.org/info/rfc5065>>.

Authors' Addresses

Enke Chen
Palo Alto Networks

Email: enchen@paloaltonetworks.com

Jenny Yuan
Palo Alto Networks

Email: jyuan@paloaltonetworks.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 5, 2021

S. Chen
Y. Zhang
H. Wang
Z. Li
Huawei Technologies
June 3, 2021

BGP Over QUIC
draft-chen-idr-bgp-over-quic-00

Abstract

Border Gateway Protocol (BGP) is an autonomous system routing protocol. The main function of BGP is to exchange routing and reachability information between autonomous systems (AS) on the Internet. BGP uses TCP to implement reliable and orderly transmission of information. Similar to TCP, QUIC is a UDP-based, byte-stream-based reliable data transmission service. In addition, by integrating with TLS 1.3, QUIC also supports functions such as establishing connections with minimum latency and providing confidentiality and integrity protection for the transmitted data, and multi-stream multiplexing. Taking use of QUIC for BGP can achieve the possible advantages. This document defines the mechanism of BGP over QUIC to and corresponding procedures.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on December 5, 2021.

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.

Table of Contents

1. Introduction	2
2. Terminology	4
3. Design Consideration	4
3.1. BGP Specification Compatibility	4
3.2. Design for Minimum Latency	4
3.3. Eliminate Head-of-line Block	4
4. Specification	5
4.1. BoQ Protocol Stack	5
4.2. Port number and ALPN	5
4.3. Stream mapping	5
4.4. BGP Session Establishment	6
4.4.1. BGP FSM	6
4.4.2. 1-RTT Handshake	10
4.4.3. 0-RTT Handshake	11
4.5. BGP session management	15
4.5.1. Error Handling	15
4.5.2. Session closure	16
5. Security Considerations	17
6. Contributors	18
7. Acknowledgments	18
8. References	18
Authors' Addresses	19

1. Introduction

BGP is used to distribute IP routes between autonomous systems and it is one of the most important Internet protocols. BGP Multiprotocol Extensions (MP-BGP) [RFC4760] enables BGP to distribute routes of various address families, such as VPN-IPv4 routes [RFC4364], VPN-IPv6 routes [RFC4659], and EVPN routes [RFC7432].

BGP is a routing protocol that requires long-term session persistence. BGP requires that transport protocol provide reliable and secure data transmission services. In [RFC4271], TCP is defined as the transport protocol for BGP. However, TCP does not protect the confidentiality of the transmitted data.

Currently, BGP uses MD5, TCP-AO and TCP Over TLS to provide integrity protection. However, MD5 has been considered an insecure encryption algorithm ([RFC6151]). TCP-AO ([RFC5925][RFC5926]) is unable to encrypt the payload. TLS ([RFC5246][RFC8446]) can be added between BGP and TCP to provide identity authentication, confidentiality, and integrity protection for BGP. However, the way is inefficient since when establishing a BGP session, a three-way handshake is adopted to establish a TCP connection, and then TLS handshake authentication is also performed.

QUIC [RFC9000] [RFC9001] is a UDP-based transport protocol that provides the following functions:

1. Reliable data transmission service based on byte streams similar to TCP.
2. Support low-latency connection establishment.
3. Authentication of the server is provided during connection establishment.
4. Authentication of the client is provided during connection establishment. (Optional)
5. QUIC provides confidentiality and integrity protection for transport data and key fields in QUIC headers. QUIC also supports periodic key updates.
6. Supports stream multiplexing, including unidirectional and bidirectional streams.
7. Supports connection migration

Comparing with BGP over TCP, BGP over Quic (BoQ) provides the following benefits:

1. The BGP session establishment delay can be reduced since the handshake times can be reduced comparing with BGP over TCP/TLS.
2. The head-of-line block between BGP address families can be eliminated by adopting stream-multiplexing for BGP.

3. Endogenous transport-layer security is provided, and no additional TLS is required.

This document defines the mechanism of BGP over QUIC to and corresponding procedures.

2. Terminology

Client: QUIC client, the active part of QUIC connection.

Server: QUIC server, the passive part of QUIC connection.

BGP over TCP: BGP using TCP as the transport layer, as [RFC4271].

BoQ: BGP over QUIC, i.e., BGP using QUIC as the transport layer.

ALPN: Application-Layer Protocol Negotiation

3. Design Consideration

3.1. BGP Specification Compatibility

BoQ replaces only the transport layer of BGP over TCP, requiring that the BGP protocol specification remain backward compatible.

Note that during the establishment of a BGP session, the BGP session state machine needs to receive transport-layer event. The BoQ also needs to receive and process QUIC-related events.

3.2. Design for Minimum Latency

QUIC provides minimal connection setup delay. The BGP session setup delay is shortened from TLS 1.3(1 RTT) + TCP(3 RTT) to QUIC(1 RTT). If a BGP session is not established for the first time, the RTT can be set to 0 to shorten the BGP session setup delay.

As the core routing protocol of the Internet, a large-scale BGP session needs to be established over a long distance. Reducing the BGP session setup delay helps improve the overall network performance.

3.3. Eliminate Head-of-line Block

Data transmitted in a BGP session can be classified into multiple objects: address family, VRF, and route prefix. Different objects can be mapped to different QUIC streams as required to isolate these objects, thereby eliminating the head-of-line blocking problem.

4. Specification

4.1. BoQ Protocol Stack

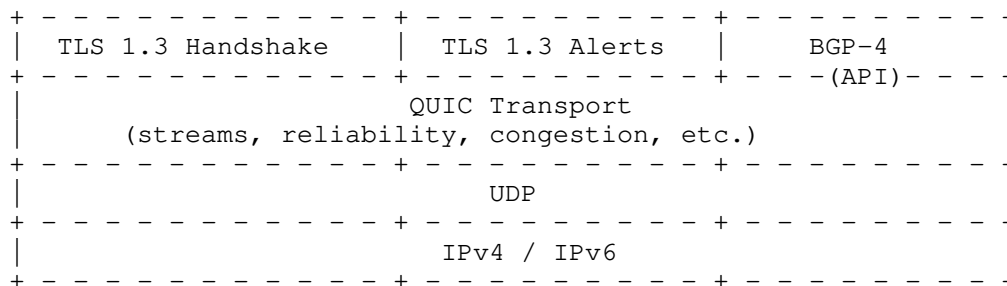


Figure 1. QUIC Protocol Stack

QUIC provides reliable data transmission services based on byte streams. In addition, QUIC uses TLS 1.3 [RFC8446] to protect confidentiality and integrity of transmitted data.

4.2. Port number and ALPN

According to Figure 1, QUIC is an application protocol that runs on top of UDP. However, QUIC is designed as a generic (application-layer) transport protocol and does not define a standalone UDP port [I-D.ietf-quic-applicability]. The QUIC uses the port of the application protocol to identify a specific application protocol. In addition, in some cases, multiple protocol versions can run simultaneously on a port number. For example, multiple protocol versions of HTTPS use TCP/UDP port 443, such as HTTP/2 [RFC7540] and HTTP/3 [I-D.ietf-quic-http].

BGP uses TCP/UDP port 179, and currently only a unique valid protocol version BGP-4 exists. Therefore, the QUIC may use only UDP port 179 to uniquely identify the BoQ (that is, BGP-4 over QUIC). The ALPN does not need to be specified.

4.3. Stream mapping

In a QUIC connection, up to $2^{62} - 1$ QUIC streams can be created and the QUIC stream can be unidirectional or bidirectional. In one connection, data of multiple streams may be transmitted at the same time. QUIC strictly ensures a transmission sequence of data of the same stream, but does not guarantee a transmission sequence of data of different streams. QUIC also supports stream-level flow control. This function is called stream multiplexing.

BGP can take use of the stream multiplexing to solve the head-of-line issues. For example, for MP-BGP [RFC4760], multiple address families can be deployed in a BGP session. Each address family can be configured with multiple VRFs, and each VRF can contain multiple route prefixes. To isolate objects at different layers and eliminate queue head blocking between these objects, the following QUIC stream mapping mode can be selected:

Option 1, Mapping streams based on address families: One or more address family can be mapped to one stream.

Option 2, Mapping streams based on VRFs: One or more VRFs can be mapped to one stream.

Option 3, Mapping streams based on prefix: it can be combinations of prefixes.

Note that regardless of which mapping mode is selected, data of the same object MUST be received and transmitted using the same QUIC stream.

4.4. BGP Session Establishment

4.4.1. BGP FSM

Before distributing routes, BGP needs to establish a point-to-point connection, called a BGP session. [RFC4271] defines BGP FSM to describe the BGP session establishment process.

A BGP session can be established in two phases:

1. Establish a transport layer connection. For BGP over TCP, a TCP connection is established through a three-way handshake. For BoQ, a 1-RTT handshake or 0-RTT handshake is used to establish a QUIC connection.

2. Establish a BGP session. After a transport-layer connection is established, BGP peers exchange BGP Open and BGP Keepalive messages. If the BGP peers reach the Established state, the BGP session has been established.

Similar to TCP, QUIC distinguishes the client (active party) from the server (passive party). Therefore, the connection conflict detection and resolution methods described in [RFC4271] are also applicable to BoQ FSM.

In this document, BGP FSM is referred to as BoQ FSM. In the 1-RTT handshake, the BoQ FSM inherits the BGP FSM defined in [RFC4271].

TCP-related session attributes should be replaced with BoQ FSM-specific session attributes and FSM events defined in this document. In addition, the processing of BoQ FSM in the 0-RTT handshake is added.

The session attributes and FSM events specific to BoQ FSM are defined as follows:

1. Session Attribute

(1) Optional Session Attributes: PassiveQuicEstablishment

Description: This option indicates that the BGP FSM will passively wait for the remote BGP peer to establish the BGP QUIC connection.

Value: TRUE or FALSE

(2) Optional Session Attributes: TrackQuicState

Description: The BGP FSM normally tracks the end result of a QUIC connection attempt rather than individual QUIC messages. Optionally, the BGP FSM can support additional interaction with the QUIC connection negotiation. .

Value: TRUE or FALSE

2. FSM Event

QUIC directly encapsulates the handshake process of TLS 1.3 [RFC8446]. In addition, QUIC requires that all packets must be explicitly acknowledged. Therefore, QUIC defines the end state of two connection establishment [RFC9001]

(1) Handshake Complete: TLS 1.3 has successfully completed the handshake.

(2) Handshake Confirmed: The QUIC has successfully completed the handshake.

On the client, the state is Handshake Complete and then Handshake Confirmed. On the server, the two states are reached at the same time.

The transport layer events for BoQ FSM are defined as follows :

Event 29: ManualStart_with_PassiveQuicEstablishment

Definition: Local system administrator manually starts the peer connection, but has PassiveQuicEstablishment enabled.

Status: Optional, depending on local system

Optional Attribute Status:

- 1) The PassiveTcpEstablishment attribute SHOULD be set to TRUE if this event occurs.
- 2) The DampPeerOscillations attribute SHOULD be set to FALSE when this event occurs.

Corresponding TCP events: Event 4

Event 30: AutomaticStart_with_PassiveQuicEstablishment

Definition: Local system automatically starts the BGP connection with the PassiveQuicEstablishment enabled.

Status: Optional, depending on local system

Optional Attribute Status:

- 1) The AllowAutomaticStart attribute SHOULD be set to TRUE.
- 2) The PassiveTcpEstablishment attribute SHOULD be set to TRUE.
- 3) If the DampPeerOscillations attribute is supported, the DampPeerOscillations SHOULD be set to FALSE.

Corresponding TCP events: Event 5

Event 31:

AutomaticStart_with_DampPeerOscillations_and_PassiveQuicEstablishment

Definition: Local system automatically starts the BGP peer connection with peer oscillation damping enabled and PassiveQuicEstablishment enabled. The exact method of damping persistent peer oscillations is determined by the implementation and is outside the scope of this document.

Status: Optional, depending on local system

Optional Attribute Status:

- 1) The AllowAutomaticStart attribute SHOULD be set to TRUE.

2) The DampPeerOscillations attribute SHOULD be set to TRUE.

3) The PassiveTcpEstablishment attribute SHOULD be set to FALSE.

Corresponding TCP events: Event 7

Event 32: QuicConnection_Valid

Definition: This parameter is applicable only to the QUIC server. It indicates that the Handshake Confirmed state is reached.

Status: Optional

Optional Attribute Status: 1) The TrackTcpState attribute SHOULD be set to TRUE if this event occurs.

Corresponding TCP events: Event 14

Event 33: Quic_CR_Invalid

Definition: This parameter applies only to the QUIC server and indicates that an invalid QUIC connection request is received. Initial packets with invalid source addresses or port numbers, invalid destination addresses or port numbers or version negotiation or address validation fails.

Status: Optional

Optional Attribute Status: 1) The TrackTcpState attribute should be set to TRUE if this event occurs.

Corresponding TCP events: Event 15

Event 34: Quic_CR_Acked

Definition: This parameter applies only to the QUIC client. It indicates that an Initial ACK message is received from the QUIC server and an Initial/Handshake message is sent to the QUIC server. Note: When this event is received, the QUIC client has reached the Handshake Complete state.

Status: Mandatory

Corresponding TCP events: Event 16

Event 35: QuicConnectionConfirmed

Definition: This parameter applies to both QUIC client and QUIC server, indicating that the Handshake Confirmed state has been reached.

Status: Mandatory

Corresponding TCP events: Event 17

Event 36: QuicConnectionFails

Definition: This parameter applies to both the QUIC client and the QUIC server. It indicates that an error occurs in the QUIC handshake before the system enters the Handshake Confirmed state.

Status: Mandatory

Corresponding TCP events: Event 18

4.4.2. 1-RTT Handshake

Normally, a BoQ should use the QUIC 1-RTT handshake to establish a BGP session because it does not require any preconditions. In particular, 1-RTT MUST be used when a BGP session is established for the first time.

When the 1-RTT handshake is used, the BoQ FSM only needs to replace the TCP event in the BGP FSM [RFC4271] with the QUIC event according to the mapping in section 4.4.1.

The QUIC has complete security only when it reaches the Handshake Confirmed state. Therefore, the BoQ FSM should allow the BGP Open message to be sent only after receiving the QuicConnectionConfirmed event.

Although to reduce the connection setup delay, QUIC allows application data to be sent before the Handshake Confirmed state is reached. However, the BGP FSM status needs to be changed for security reasons and the same issues as 0-RTT. Therefore, it is not recommended that the BGP Open message be sent before the Handshake Confirmed state is reached.

If the DelayOpen or PassiveQuicEstablished function is configured on the local system, 1-RTT is also required.

4.4.3. 0-RTT Handshake

When the 0-RTT handshake is used, the QUIC client sends a connection establishment request (Initial packet) with a BGP Open Data message. (Referred to as early data and sent using 0-RTT packets) , which means:

- (1) After sending an Initial packet, the client enters the BGP OpenSent state.
- (2) After receiving the Client Initial packet and sending the Server Initial/Handshake packet, the server may send BGP Open and change the status to BGP OpenConfirmed. At this time, the Server has not reached the Handshake Complete state.
- (3) When receiving the Server Initial/Handshake/BGP Open message, the client also reaches the BGP OpenConfirmed state. In this case, the client is not in the Handshake Complete state either.

Therefore, in the 0-RTT handshake, the BoQ FSM can directly skip the BGP Connect and Active states. This minimizes the BGP session setup delay. After a BGP peer is disconnected from the Established state, 0-RTT can be used for re-establishment.

It should be noted that when the BoQ reaches the BGP OpenConfirmed state, because neither the client nor the server reaches the Handshake Complete state, the handshake may fail. Therefore, during the 0-RTT handshake, before the Handshake Confirmed state is reached, that is, before the BoQ FSM receives the QuicConnectionConfirmed event, the BGP KEEPALIVE/UPDATE/ROUTE-REFRESH message MUST NOT be sent but the NOTIFICATION message MAY be sent.

When the 0-RTT handshake is used to establish a BGP session, delete the Connect and Active states and replace Section 8.2.2 of [RFC4271] with the following content.

For details about the event indexes, refer to [RFC4271].

Idle state:

Initially, the BGP peer FSM is in the Idle state. Hereafter, the BGP peer FSM will be shortened to BGP FSM.

In this state, BGP FSM refuses all incoming BGP connections for this peer. No resources are allocated to the peer. In response to a ManualStart event (Event 1) or an AutomaticStart event (Event 3), the local system:

- initializes all BGP resources for the peer connection,
- sets ConnectRetryCounter to zero,
- starts the ConnectRetryTimer with the initial value,
- initiates a QUIC connection to the other BGP peer, In addition, the Initial packet carries the BGP Open packet.
- listens for a connection that may be initiated by the remote BGP peer, and
- changes its state to OpenSent.

For details about how to handle other events in this state, refer to [RFC4271].

OpenSent:

In this state, the BGP OPEN packet has been sent to the neighbor along with the Initial packet.

The start events (Events 1, 3, 6, 29-31) are ignored in the OpenSent state.

If a ManualStop event (Event 2) is issued in the OpenSent state, the local system:

- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the QUIC connection,
- sets the ConnectRetryCounter to zero, and
- changes its state to Idle.

In response to the ConnectRetryTimer_Expires event (Event 9), the local system:

- drops the QUIC connection,
- restarts the ConnectRetryTimer,
- initiates a QUIC connection to the other BGP peer. In addition, the Initial packet carries the BGP Open packet.

- continues to listen for a connection that may be initiated by the remote BGP peer, and
- stays in the OpenSent state.

If the local system receives a DelayOpenTimer_Expires event (Event12), the local system:

- sends the NOTIFICATION with the Error Code Finite State Machine Error,
- sets the ConnectRetryTimer to zero,
- stops and clears the DelayOpenTimer (set to zero),
- drops the QUIC connection, - increments the ConnectRetryCounter by 1,
- optionally performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to IDLE.

If the BGP FSM receives a QuicConnection_Valid event (Event 32), the QUIC connection is processed, and the connection remains in the OpenSent state.

If the BGP FSM receives a Quic_CR_Invalid event (Event 15), the local system rejects the QUIC connection, and the connection remains in the Opensent state.

If the QUIC connection succeeds (Event 34 or Event 35), the local system has sent the OPEN packet to the neighbor, the local system:

- stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,
- sets the HoldTimer to a large value, and
- stays in the OpenSent state.

If the TCP connection fails (Event 36), the local system:

- restarts the ConnectRetryTimer (with the initial value),
- releases all BGP resources,
- increments the ConnectRetryCounter by 1,

- optionally performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE,
- continues to listen for a connection that may be initiated by the remote BGP peer, and
- changes its state to Idle.

If the BGP message header checking (Event 21) or OPEN message checking detects an error (Event 22) (refer to Section 6.2 of [RFC4271]), the local system:

- sends a NOTIFICATION message with the appropriate error code,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the QUIC connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is TRUE, and
- changes its state to Idle.

If a NOTIFICATION message is received with a version error (Event 24), the local system:

- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the QUIC connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is TRUE, and
- changes its state to Idle.

When an OPEN message is received, all fields are checked for correctness. If there are no errors in the OPEN message (Event 19), the local system:

- sets the BGP ConnectRetryTimer to zero,

- sends a KEEPALIVE message, and
- sets a KeepaliveTimer (via the text below)
- sets the HoldTimer according to the negotiated value (refer to Section 4.2 of [RFC4271]),
- changes its state to OpenConfirm.

In response to any other event (Events 8,10-11,20,23,25-28),the local system:

- sends the NOTIFICATION with the Error Code Finite State Machine Error,
- sets the ConnectRetryTimer to zero,
- releases all BGP resources,
- drops the QUIC connection,
- increments the ConnectRetryCounter by 1,
- (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
- changes its state to Idle.

OpenConfirm:

For details, refer to [RFC4271] . The only modification is to replace TCP Event with QUIC Event. For details, refer to section 4.4.1.

Established:

For details, refer to [RFC4271]. The only modification is to replace TCP Event with QUIC Event. For details, refer to section 4.4.1.

4.5. BGP session management

4.5.1. Error Handling

As shown in section 4.4.1, BoQ error handling involves the following three types of errors:

(1) QUIC error: Includes stream error and connection error [RFC9001]. In some cases, a stream error may cause a connection error. For

example, if an operation error occurs on all streams, the connection error should be triggered to close the connection.

(2) TLS alert: In [RFC9001], a QUIC endpoint MUST treat any alert from TLS as if it were at the "fatal" level. For TLS alerts, this includes replacing any alert with a generic alert, such as `handshake_failure` (0x128 in QUIC).

(3) BGP error: If an error occurs in BGP processing [RFC4271], it can be mapped to the following BoQ Error Codes [RFC9000].

This document defines some of the following BoQ Error Codes:

(1) `BOQ_NO_ERROR` (0x00): No error. This is used when the connection or stream needs to be closed, but there is no error to signal.

(2) `BOQ_INTERNAL_ERROR` (0x01): The BoQ implementation encountered an internal error and is incapable of continuing the stream or the connection.

4.5.2. Session closure

QUIC provides three ways to close a connection (refer to Section 10 of [RFC9000]):

(1) Idle timeout

(2) Immediate Close

(3) Stateless Reset

When the idle timer expires, the connection is closed immediately. Idle timeout can be calculated using the following formula:

$$\text{idle_timeout} = \text{MAX}(\text{min_idle_timeout}, 3 \cdot \text{PTO})$$

The PTO is a time that the sender should wait for an acknowledgment of a sent packet. For a calculation method, refer to Section 6.2.1 of [RFC9002] .

When establishing a QUIC connection, the transmission parameter `max_idle_timeout` is used. Endpoints advertise local `idle_timeout` to each other. If no `max_idle_timeout` advertisement is received from the remote end, the remote `idle_timeout` is set to a value of 0. Based on the values of local `idle_timeout` and remote `idle_timeout`, there are three possible scenarios:

(1) If both the values are 0, disable the idle timeout function.

(2) If there is only one value 0, set `min_idle_timeout` to a non-zero value in between.

(3) If neither value is 0, set `min_idle_timeout` to the smaller value.

Two options are available for the idle timer during BGP session establishment. Option 1 is recommended by default.

Option 1: Set this parameter to 0, indicating that idle timeout is disabled.

Option 2: The value must be greater than the value of BGP HoldTimer. It is recommended that the value be greater than five times the value of BGP HoldTimer.

5. Security Considerations

This document replaces the transport protocol layer of BGP from TCP to QUIC. It does not modify the basic protocol specifications of BGP, and therefore does not introduce new security risks to the basic BGP protocol.

BoQ enhances transport-layer security for BGP sessions, refer to [RFC7475] :

- (1) Supports server identity authentication.
- (2) (Optional) Supports client identity authentication.
- (3) Confidentiality protection of BGP messages is supported. All BGP messages are encrypted for transmission.
- (4) Supports integrity protection for BGP messages.

As described in Section 8 of [RFC8446] and Section 9.2 of [RFC9001], the 0-RTT handshake may cause replay attacks. To avoid Replay attacks, the following methods are recommended:

- (1) By default, the 0-RTT handshake is not used to establish a BGP session.
- (2) If the 0-RTT handshake is used to establish a BGP session, it is recommended that the receiver directly discard the replayed packets in case of replay attacks. This does not affect the BGP session establishment. RFC 8470 also provides detailed analysis and mitigation measures for the risk of replay attacks caused by the use of early TLS data.

6. Contributors

TBD

7. Acknowledgments

TBD.

8. References

- [I-D.ietf-quic-applicability]
Kuehlewind, M. and B. Trammell, "Applicability of the QUIC Transport Protocol", draft-ietf-quic-applicability-11 (work in progress), April 2021.
- [I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-34 (work in progress), February 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", RFC 4659, DOI 10.17487/RFC4659, September 2006, <<https://www.rfc-editor.org/info/rfc4659>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.

- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.
- [RFC7475] Dawkins, S., "Increasing the Number of Area Directors in an IETF Area", BCP 9, RFC 7475, DOI 10.17487/RFC7475, March 2015, <<https://www.rfc-editor.org/info/rfc7475>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <<https://www.rfc-editor.org/info/rfc9000>>.
- [RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <<https://www.rfc-editor.org/info/rfc9001>>.
- [RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <<https://www.rfc-editor.org/info/rfc9002>>.

Authors' Addresses

Shuanglong Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: chenshuanglong@huawei.com

Yongkang Zhang
Huawei Technologies
101 Yuhuatai Software Avenue
Nanjing
China

Email: zhangyongkang@huawei.com

Haibo Wang
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: rainsword.wang@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 June 2022

H. Chen
M. McBride
Futurewei
R. Chen
ZTE Corporation
G. Mishra
Verizon Inc.
A. Wang
China Telecom
Y. Liu
China Mobile
Y. Fan
Casa Systems
B. Khasanov
Yandex LLC
L. Liu
Fujitsu
X. Liu
Volta Networks
19 December 2021

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 briefs 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.

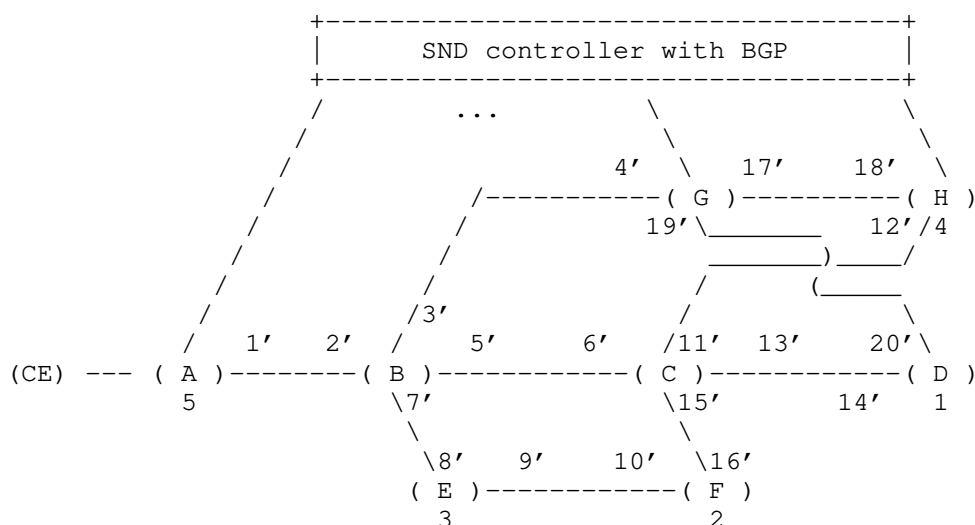


Figure 1: Example BIER-TE Topology with Controller

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.

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.

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:

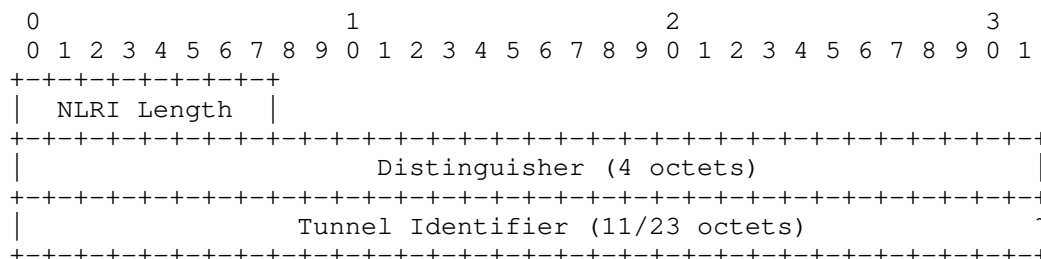


Figure 2: NLRI Format

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.

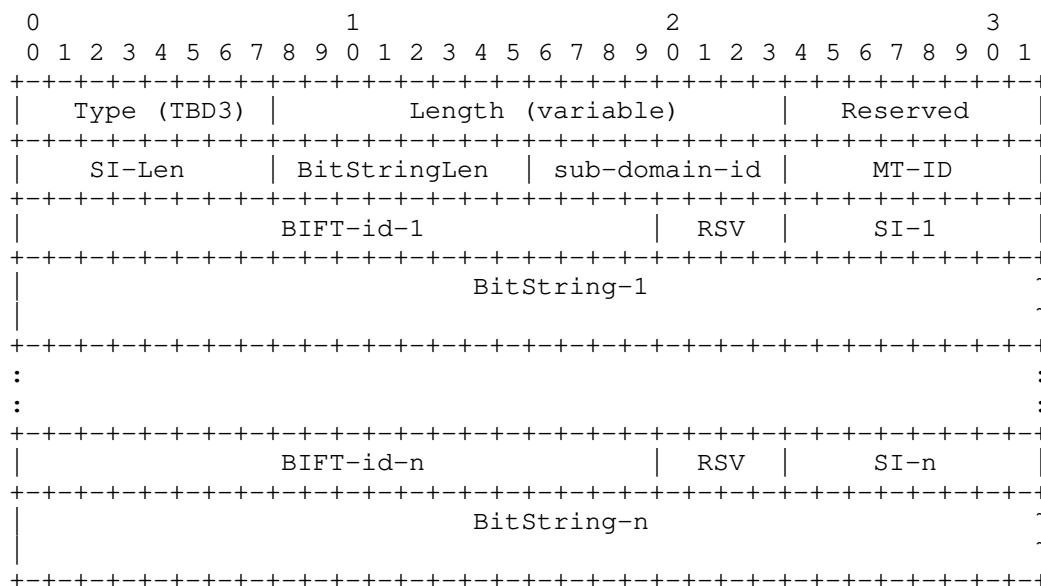


Figure 3: Path BitPositions Sub-TLV Format

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.

BitStringLength (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 BitStringLength is $\log_2(k)-5$. For example, BitStringLength = 1 indicates $k = 64$, BitStringLength = 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.

<BIFT-id, SI, BitString> tuple: Each tuple <BIFT-id- i , SI- i , BitString- i > ($i = 1, 2, \dots, n$) 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.

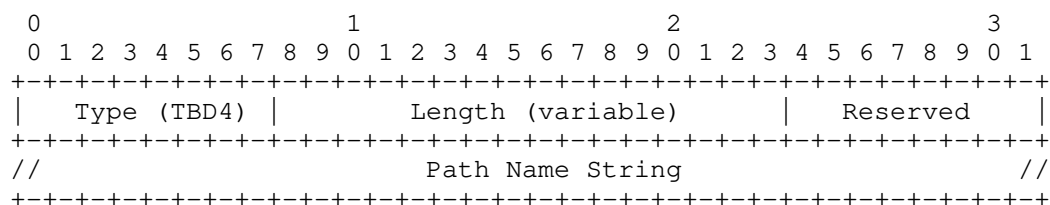


Figure 4: Path Name Sub-TLV Format

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.

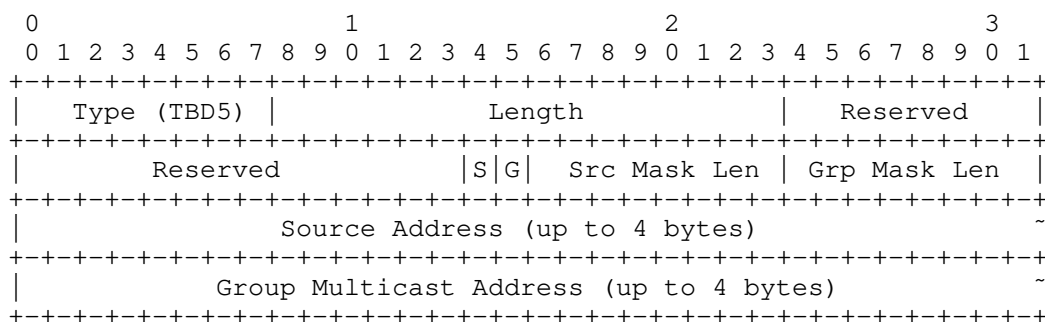


Figure 5: Multicast Traffic for IPv4 Sub-TLV

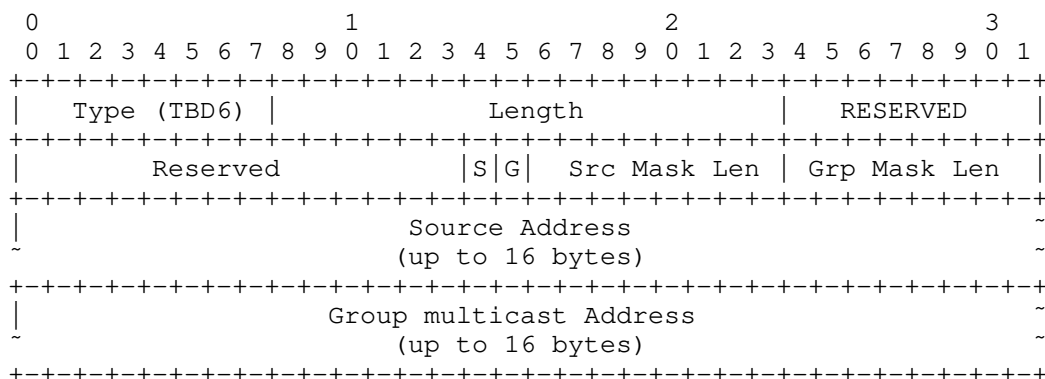


Figure 6: Multicast Traffic for IPv6 Sub-TLV

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:

Code Point	Description	Reference
TBD1 (179 suggested)	BIER-TE Policy SAFI	This document

6.2. 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:

Code Point	Description	Reference
TBD2 (16 suggested)	BIER-TE Tunnel/Path	This document

6.3. 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:

Code Point	Description	Reference
TBD3 (16 suggested)	Path BitPositions	This document
TBD4 (17 suggested)	Path Name	This document
TBD5 (18 suggested)	IPv4 Multicast Traffic	This document
TBD6 (19 suggested)	IPv6 Multicast Traffic	This document

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <<https://www.rfc-editor.org/info/rfc6514>>.

- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8279] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast Using Bit Index Explicit Replication (BIER)", RFC 8279, DOI 10.17487/RFC8279, November 2017, <<https://www.rfc-editor.org/info/rfc8279>>.
- [RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <<https://www.rfc-editor.org/info/rfc8296>>.
- [RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <<https://www.rfc-editor.org/info/rfc9012>>.

7.2. Informative References

- [I-D.ietf-bier-te-arch]
Eckert, T., Cauchie, G., and M. Menth, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, draft-ietf-bier-te-arch-11, 15 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-11.txt>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.

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.

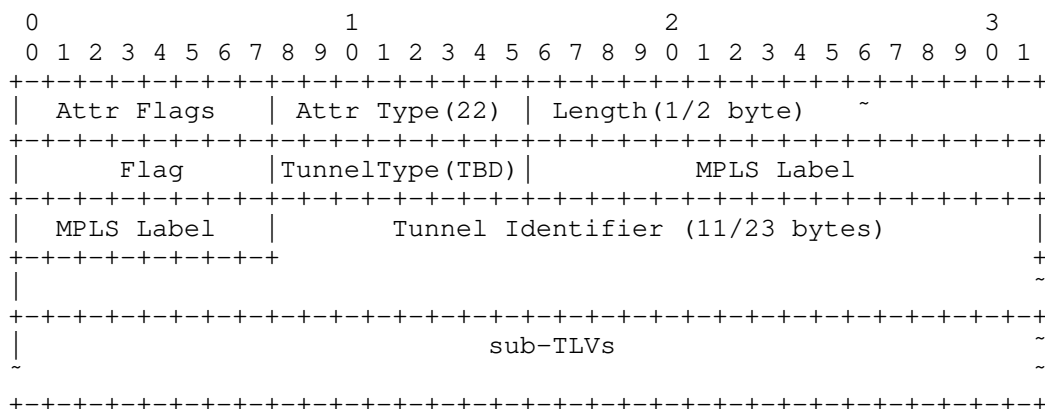


Figure 7: PTA with Tunnel Type for BIER-TE

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

- o Tunnel Type: It is set to be TBD, indicating BIER-TE tunnel.
- o Tunnel Identifier: It contains: sub-domain-id of 1 byte, BIER-TE tunnel BFIR's BFR-id of 2 bytes, Tunnel-ID of 4 bytes, and BIER-TE tunnel BFIR's BFR-prefix of 4/16 bytes for IPv4/IPv6.
- o 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.
- o Others: The other fields are set according to [RFC6514].

Authors' Addresses

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

RTG Working Group
Internet Draft
Intended status: Standard track
Expires: October 10, 2022

L. Dunbar
Futurewei
K. Majumdar
CommScope
U. Chunduri
Intel
July 10, 2021

BGP Dissemination of FlowSpec for Transport Aware Mobility
draft-dmc-idr-flowspec-tn-aware-mobility-01

Abstract

This document defines a BGP Flow Specification (flowSpec) extension to disseminate flows from 5G mobile networks so that the 5G mobile systems slices and Service Types (SSTs) can be mapped to optimal underlying network paths in the data network outside the 5G UPFs, or the N6 interface in 3GPP 5G Architecture [3GPP TR 23.501].

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 23, 2021.

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

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	3
3. TN-Aware matching conditions.....	4
4. Redirect a flow over an underlay tunnel.....	6
5. FlowSpec Redirect to Indirection-ID Non-Transitive Extended Community.....	8
6. IANA Considerations.....	9
7. Security Considerations.....	9
8. Contributors.....	9
9. References.....	9
9.1. Normative References.....	9
9.2. Informative References.....	10
10. Acknowledgments.....	10
Authors' Addresses.....	12

1. Introduction

The [TN-AWARE-MOBILITY-EXT] describes a framework for extending the mobility aware transport network characteristics through the Data Network outside the 5G UPFs.

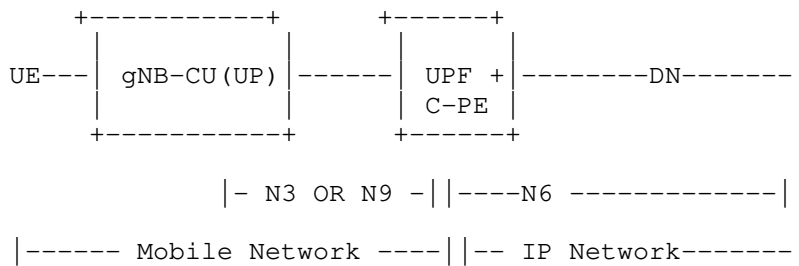


Figure 1: Mobile and IP Data Network for UE

The 5G UPF terminates the 5G GTP tunnels from gNB and pass the IP packets to the N6 data networks, which deliver the packets over hybrid paths, like MPLS, SR paths, Private-IP, or public Internet to reach the packets' destinations.

This document focuses on using FlowSpec to disseminate rules that utilize the mobility aware transport network characteristics to forward 5G flows.

Border Gateway Protocol (BGP) Flow Specification (FlowSpec) [RFC8955] and FlowSpec for IPv6 [RFC8956] leverage the BGP Control Plane to simplify the distribution of rules for the specified flows. FlowSpec filter rules can be injected to all BGP peers simultaneously without changing router configuration.

2. Conventions used in this document

BSID	- Binding SID
DC	- Data Center
DN	- Data Network (5G)
EMBB	- enhanced Mobile Broadband (5G)
gNB	- 5G NodeB

GTP-U	- GPRS Tunneling Protocol - Userplane (3GPP)
MIOT	- Massive IOT (5G)
PECP	- Path Computation Element (PCE) Communication Protocol
SD-WAN	- Software-Defined Wide Area Network
SID	- Segment Identifier
SLA	- Service Layer Agreement
SST	- Slice and Service Types (5G)
SR	- Segment Routing
SR-PCE	- SR Path Computation Element
UE	- User Equipment
UPF	- User Plane Function (5G)
URLLC	- Ultra reliable and low latency communications (5G)

3. TN-Aware matching conditions

[RFC8955] defines a BGP Network Layer Reachability Information (NLRI) format used to distribute traffic flow specification rules. The NLRI for (AFI=1, SAFI=133) specifies IPv4 unicast filtering. The NLRI for (AFI=1, SAFI=134) specifies IPv4 BGP/MPLS VPN filtering [RFC7432]. The Flow Specification match part defined in [RFC8955] includes L3/L4 information like IPv4 source/destination prefix, protocol, ports, and the like, so traffic flows can be filtered based on L3/L4 information. This has been extended by [RFC8956] to cover IPv6 (AFI=2) L3/L4.

The NLRI FlowSpec components described in RFC8955 and RFC8956 are adequate for specifying the UDP Source Port Range which is used to differentiate SLAs of flows from UPFs [EXT-TN-AWARE-Mobility].

The Ingress PE, which is either a function inside UPF or directly connected to UPF, acting as BGP FlowSpec Receiver is assumed to have a BGP FlowSpec session with the FlowSpec Controller. The Mobility traffic destination would resolve in the BGP Peer Next Hop in the data network. The BGP FlowSpec Controller would be programmed with {5G UDP Src Port Range} to map different SSTs defined in [TN-AWARE-MOBILITY] to create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP FlowSpec Generalized Indirection-ID}. The Mobility IP packets coming out of the UPF, i.e., GTP header being decapsulated, carrying specific UDP Source Port can be classified based on the matching policy carried by the FlowSpec NLRI.

For example, to filter out flows with source UDP port number between [i, j], the following encoding can be used in the NLRI (SAFI=133 or SAFI 134):

Encoding

<Type = 6, [numeric_op1, i][numeric_op2, j]>

<Type = 2, [numeric_op3, Src-Prefix]>

<Type = 1, [numeric_op4, Dest-prefix]>

Numeric_Op1 is:

0	1	2	3	4	5	6	7
e	a	len		0	lt	gt	eq
0	1	00		0	0	1	0

Numeric_Op2 is:

0	1	2	3	4	5	6	7
e	a	len		0	lt	gt	eq
1	1	00		0	1	0	0

Where len ==0, meaning two bytes of value [i] follows the Numeric_op1 and two bytes of value [j] follows the Numeric_op2.

The "numeric_op3" and "numeric_op4" are for comparing the source and destination addresses of the UE traffic.

4. Redirect a flow over an underlay tunnel

For the flows matching with the filter conditions carried by the FlowSpec NLRI, the policy for redirect path can indicate a set of underlay tunnels or one underlay tunnel.

As the action of taking specific underlay tunnels is performed by the headend router, a non-transitive Extended Community for Path Redirect [Flowspec-path-redirect] and [SRv6-flowspec-path-redirect] should be used.

[IANA Action: need a new type:

0x49 FlowSpec Redirect to Indirection-id Non-transitive
Extended Community.

]

For hierarchical RR deployments where the FlowSpec rules need to be propagated, the Transitive Path Redirect Extended Community [FlowSpec-path-redirect] can be used.

The below figure tries to capture the overall topology, showing the mobility traffic from UPF being redirected to different paths per the BGP FlowSpec from the Controller:

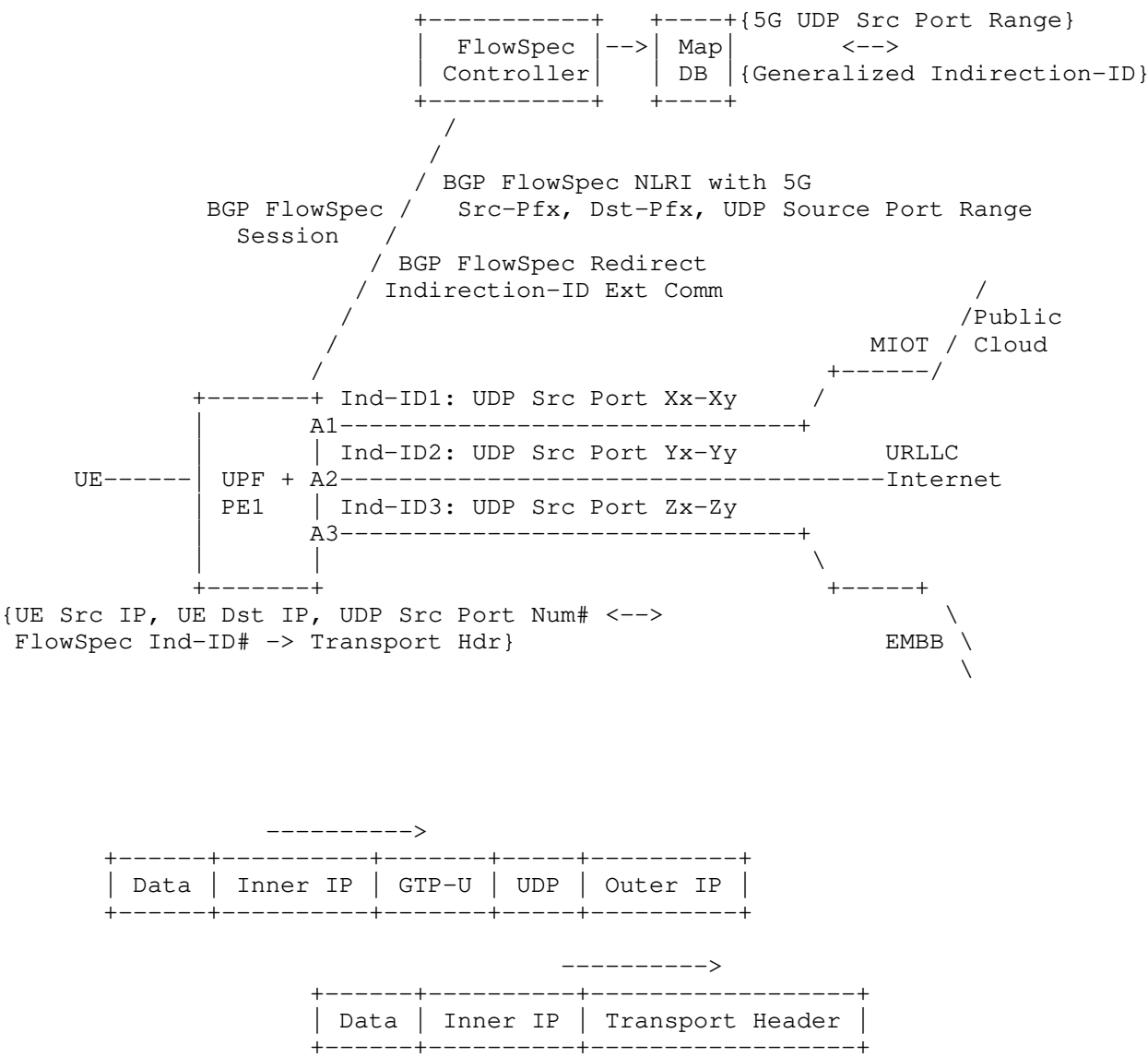


Figure 2: TN Aware Mobility Traffic Mapping to FS Redirect Path

5. FlowSpec Redirect to Indirection-ID Non-Transitive Extended Community

This section defines "FlowSpec Redirect to Indirection-ID Non-Transitive Extended Community for IPsec Tunnel ID". The format of this extended community is shown below:

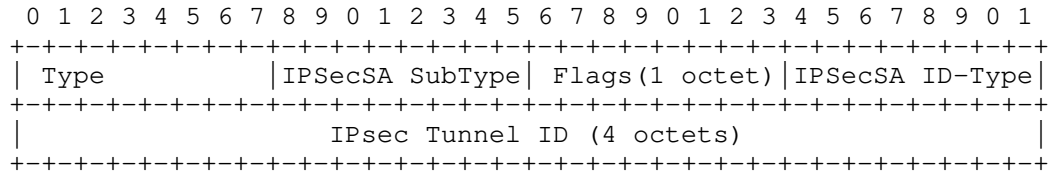


Figure 3: Redirect to Ind-ID Ext Community for IPsec Tunnel

Where

Type = 0x49 (to be assigned by IANA): Non-Transitive FlowSpec Redirect to Indirection-ID Extended Community for IPsec Tunnel ID.

[Note: Type = 0x09 for Transitive FlowSpec Redirect to Indirection-ID Extended Community can also be used for Hierarchical deployment, where the FlowSpec Update needs to be propagated]

IPsec SA Sub-Type: 1 octet, its value (TBD) will be assigned by IANA to indicate the ID carried by the Extended Community is IPsec SA ID. Assuming the IPsec SA is pre-established, its Security Association (SA) ID is within a single administrative domain a globally unique identifier. The allocation and establishment of the IPsec SA among peers is outside scope of the document.

Flags: Same as that defined in [Flowspec-path-redirect].

IPsec SA ID-Type: 1 octet value. Here is the new value needed for IPsec IPv4 tunnel (to be assigned by IANA)

v1 - Inner Encap type = IPsec+GRE

v2 - Inner Encap type = IPSec+Vxlan

6. IANA Considerations

This draft needs an IANA code point allocation for the Non-Transitive FlowSpec Redirect to Indirection-ID Extended Community.

Type: Non-Transitive FlowSpec Redirect to Indirection-ID
Extended Community for IPSec Tunnel ID.

IPsec SA Sub-Type:

IPSec SA ID-Type:

v1 - Inner encap type = IPSec+GRE

v2 - Inner encap type = IPSec+Vxlan

7. Security Considerations

TBD.

8. Contributors

The following people have contributed to this document.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC8955] C. Loibl, et al, "Dissemination of Flow specification Rules", Dec 2020.

[RFC8956] C. Loibl, et, al, "Dissemination of Flow Specification Rules for IPv6". Dec 2020.

9.2. Informative References

[RFC5440] JP. Vasseur, Ed., JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", March 2009

[Flowspec-path-redirect] G. Van De Velde, et al, "Flowspec Indirection-id Redirect", draft-ietf-idr-flowspec-path-redirect-11, March 2020

[SRv6-Flowspec-path-redirect] G. Van De Velde, et al, "Flowspec Indirection-id Redirect for SRv6", draft-ietf0-idr-srv6-flowspec-path-redirect-05, Jan. 2021

[TN-AWARE-MOBILITY] U. Chunduri, et al, "Transport Network aware Mobility for 5G", draft-clt-dmm-tn-aware-mobility-07, April 2021

[TN-AWARE-MOBILITY-EXT] K. majumdar, et al, "Extension of Transport Aware Mobility in Data Network", draft-mcd-rtgwg-extension-tn-aware-mobility-01, May 2021

[BGP-SR-TE-POLICY] S. Previdi, et al, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-09, November 2020

[SDWAN-BGP-USAGE] L. Dunbar, et al, "BGP Usage for SDWAN Overlay Networks", draft-dunbar-bess-bgp-sdwan-usage-08, January 2021

[SDWAN-Edge-Discover] L. Dunbar, et al, "BGP UPDATE for SDWAN Edge Discovery", draft-dunbar-idr-sdwan-edge-discovery-04, April 2021

10. Acknowledgments

TBD.

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Linda Dunbar
Futurewei
2330 Central Expressway
Santa Clara, CA 95050

Email: linda.dunbar@futurewei.com

Kausik Majumdar
CommScope
350 W Java Drive, Sunnyvale, CA 94089

Email: kausik.majumdar@commscope.com

Uma Chunduri
Intel
2200 Mission College Blvd
Santa Clara, CA 95052

Email: umac.ietf@gmail.com

BESS WorkGroup
Internet-Draft
Intended status: Standards Track
Expires: October 29, 2022

D. Rao
S. Agrawal
C. Filsfils
Cisco Systems
D. Steinberg
Lapishills Consulting Limited
L. Jalil
Verizon
Y. Su
Alibaba, Inc
B. Decraene
Orange
J. Guichard
Futurewei
K. Talaulikar
K. Patel
Arrcus, Inc
H. Wang
Huawei Technologies
April 27, 2022

BGP Color-Aware Routing (CAR)
draft-dskc-bess-bgp-car-04

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

	some resources.
Colored Service Route	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].
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	<p>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.</p> <p>Service Steering: Service route -> BGP CAR path (or other Color-Aware Routed Paths: e.g., SR Policy)</p> <p>Intra-Domain Resolution: BGP CAR route -> intra-domain color aware path (e.g. SR Policy, IGP Flex-Algo, BGP CAR)</p>

1.2. Illustration

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

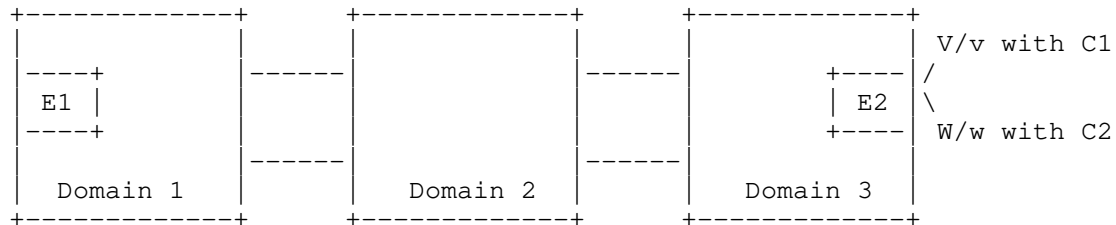


Figure 1

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

- o C1 is mapped to "low-delay"
 - * Flex-Algo FA1 is mapped to "low delay" and hence to C1
- o 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:

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

E1 has the following color-aware paths:

- o (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

- o (E2, C2) provided by SR Policy

E1 automatically steers the received service routes as follows:

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

Illustrated Properties:

- o Leverage of the BGP Color Extended-Community
 - * The service routes are colored with widely-used BGP Extended-Color Community
- o (E, C) Automated Steering
 - * V/v and W/w are automatically steered on the appropriate color-aware path
- o 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
- o 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:

- o 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.
- o Control-Plane: a node should not install a (E, C) path if it does not need it
- o 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:

- o NLRI Route-Type field: provides extensibility to add new NLRI formats for new route-types
- o Key length: field enables handling of unsupported route-types opaquely, enabling transitivity via RRs
- o TLV-based encoding of non-key NLRI: enables support for multiple encapsulations with efficient update packing
- o AIGP Attribute provides extensibility via TLVs, enabling definition of additional metric semantics for a color as needed for an intent

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:

- o the color constraint in the first check may be relaxed: instead N is reachable in the default routing table
- o the dataplane availability constraint of T may be relaxed
- o 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

```
[Color Domain 1  A]-----[B      Color Domain 2      E2]
[C1=low-delay    ]        [C2=low-delay                ]
```

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:

- o Within domain 2, the BGP CAR route is (E2, C2) via E2
- o B signals to A the BGP CAR route as (E2, C2) via B with Local-Color-Mapping-Extended-Community (LCM-EC) of color C2
- o A is aware (classic peering agreement) of the intent-to-color mapping within domain 2 ("low-delay" in domain 2 is C2)
- o 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)
- o 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:

- o The NLRI never changes
- o E is globally unique, which makes E-C in that order unique
- o In the vast majority of the case, the color of the NLRI is used for resolution and steering
- o 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									
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
NLRI Length										Key Length										NLRI Type																			
										Type-specific Key Fields																													
										Type-specific Non-Key Fields (if applicable)																													

where:

- o NLRI Length: 1 octet field that indicates the length in octets of the NLRI excluding the NLRI Length field itself.
- o 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.
- o NLRI Type: 1 octet field that indicates the type of the BGP CAR NLRI.
- o Type-Specific Key Fields: Depend on the NLRI type and of length indicated by the Key Length.
- o 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:

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
| NLRI 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:

- o NLRI Length: variable
- o 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.
- o NLRI Type: 1

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

```

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

```

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

```

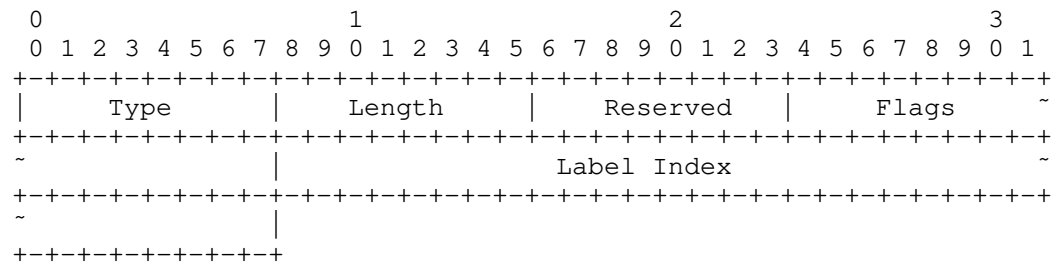
where:

- o Type : Type code is 1. T bit MUST be unset
- o Length: variable, MUST be a multiple of 3
- o 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:



where:

- o Type : Type code is 2. T bit MUST be set
- o Length: 7
- o Reserved: 1 octet field that MUST be set to 0 and ignored on receipt.
- o Flags: 2 octet field that maps to the Flags field of the Label-Index TLV of the BGP Prefix SID Attribute [RFC8669].
- o 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 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|          Type          |      Length      | SRv6 SID Info (variable)  //
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

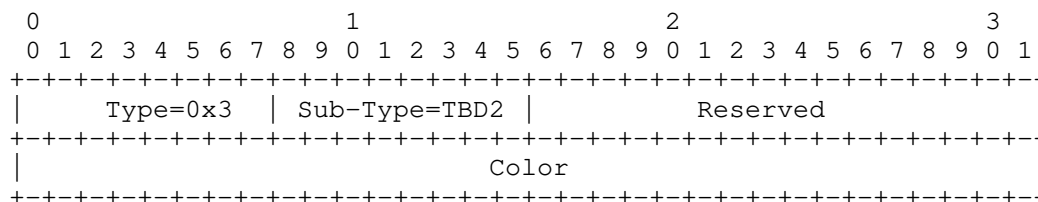
where:

- o Type : Type code is 3. T bit MUST be unset
- o Length: variable, MUST be either less than or equal to 16, or be a multiple of 16
- o 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:

- o Type: 0x3
- o Sub-Type: TBD2.
- o Reserved: 2 octet of reserved field that MUST be set to zero on transmission and ignored on reception.
- o 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':

- o Minimum NLRI length check error.
- o 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

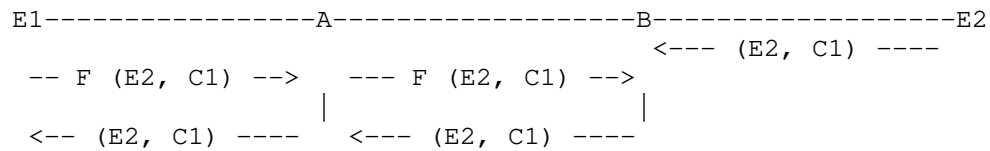
It is straightforward to note that the BGP CAR color-aware alternative supports intents 1, 2, 4 and 7.

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



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

future version of this document

6. Scaling

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

- o No intermediate node dataplane should need to scale to (Colors * PEs)
- o 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

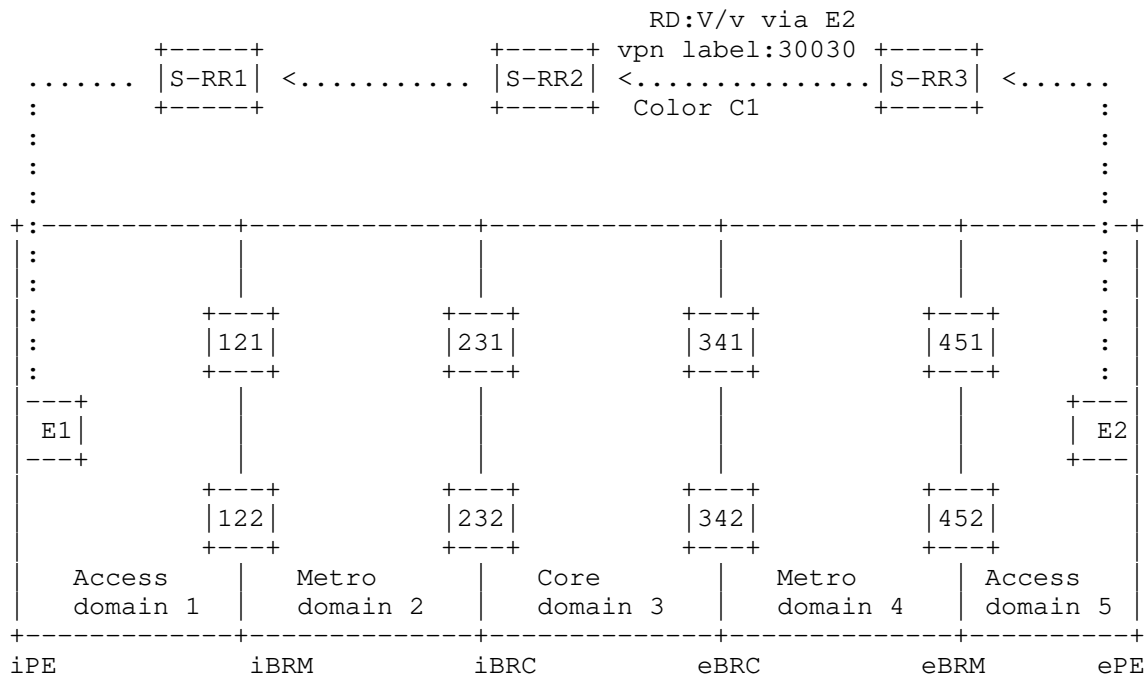


Figure 2: Ultra-Scale Reference Topology

The following applies to the reference topology above:

- o Independent ISIS/OSPF SR instance in each domain.
- o Each domain has Flex Algo 128. Prefix SID for a node is SRGB 168000 plus node number.
- o A BGP CAR route (E2, C1) is advertised by egress BRM node 451. The route is sourced locally from redistribution from IGP-FA 128.
- o Not shown for simplicity, node 452 will also advertise (E2, C1).
- o When a transport RR is used within the domain or across domains, ADD-PATH is enabled to advertise paths from both egress BRs to it's clients.
- o 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.

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

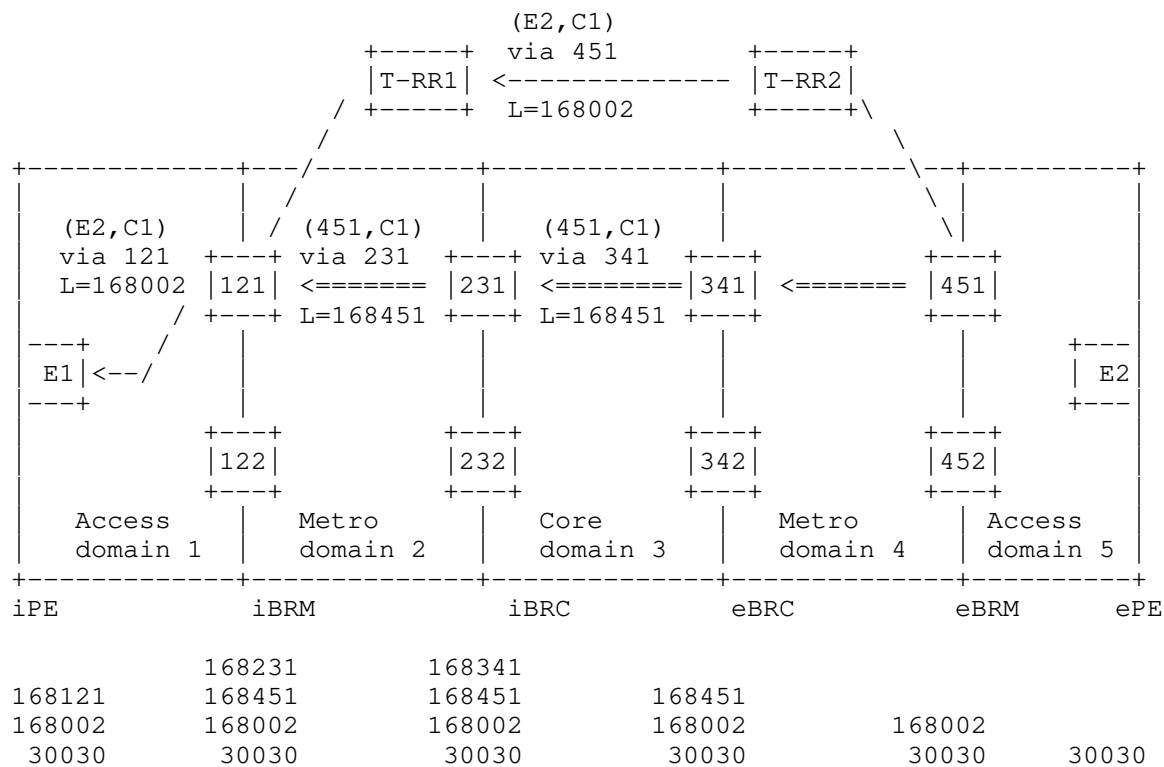


Figure 4: Heirarchical BGP transport CAR, NHS at iBR

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

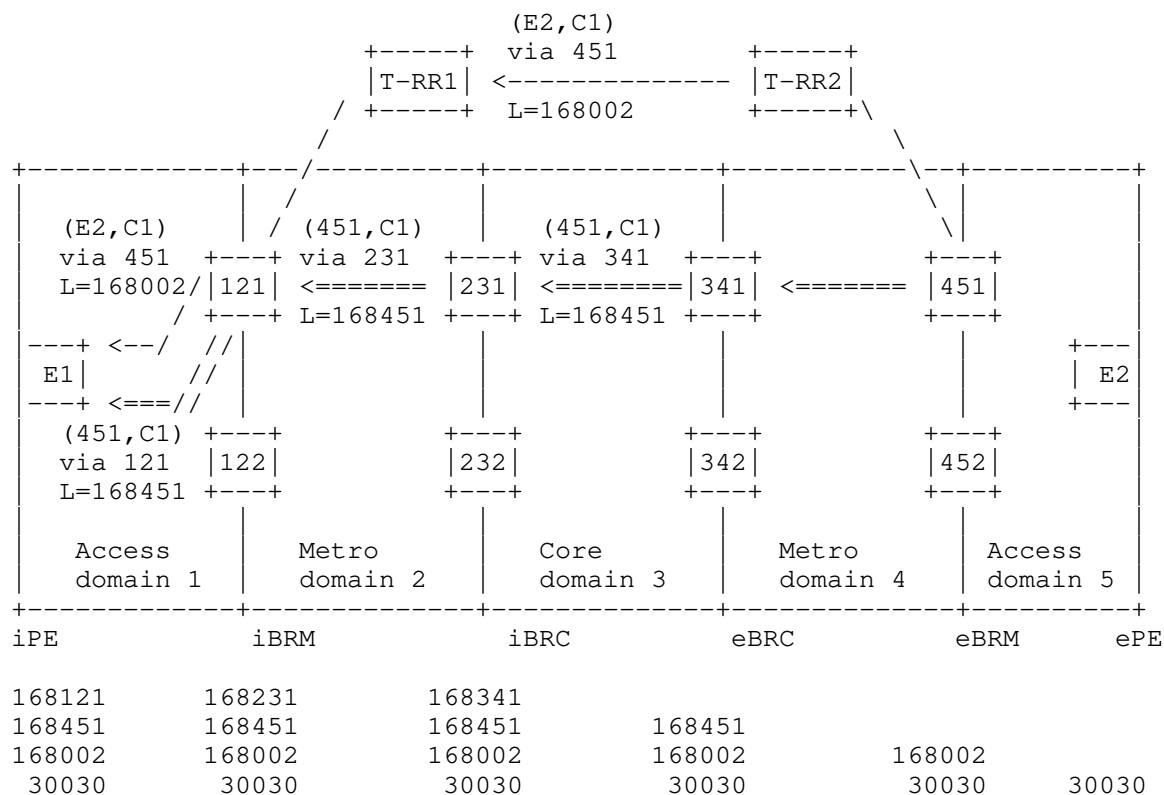


Figure 5: Heirarchical BGP transport CAR, NHU at iBR

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:

	E1	121	231
FLAT	(E2,C) via (121,C)	(E2,C) via (231,C)	(E2,C) via (341,C)
H.NHS	(E2,C) via (121,C)	(E2,C) via (451,C) (451,C) via (231,C)	(451,C) via (341,C)
H.NHU	(E2,C) via (451,C) (451,C) via (121,C)	(451,C) via (231,C)	(451,C) via (341,C)
	E1	121	231
FLAT	V -> 30030 168002 168121	168002 -> 168002 168231	168002 -> 168002 168341
H.NHS	V -> 30030 168002 168121	168002 -> 168002 168451 168231	168451 -> 168451 168341
H.NHU	V -> 30030 168002 168451 168121	168451 -> 168451 168231	168451 -> 168451 168341

- o 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.

- o 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

- o An ingress PE (E1) only learns (E, C) routes that it needs to install into data plane for service route automated steering
- o 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 its ingress PEs (inline RR with Next-Hop-Unchanged)
- o 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
- o 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

- o 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.
- o 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.

- o 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.
- o 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

- o BGP CAR is enabled between CE1-PE1 and PE2-CE2
 - o BGP VPN CAR is enabled between PE1 and PE2
 - o Provider publishes intent 'low-delay' is mapped to color CP on its inbound peering links
 - o Within its infrastructure, Provider maps intent 'low-delay' to color CPT
 - o 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)
- / connected OI
- 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 C
- olor Extended
- Community (CPT
-)
4. PE1 installs in VRF A: [(V, CC), L2] via (PE2, CPT) steered on (PE
- 2, 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
- M
6. CE1 installs : [(V, CC), L3] via PE1 which resolves on (PE1, CC)
- / connected OI
- 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							
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
NLRI 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:

Type	NLRI Type	Reference
0	Reserved (not to be used)	[This document]
1	Color-Aware Routes NLRI	[This document]
2-255	Unassigned	

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:

Type	NLRI Type	Reference
0	Reserved (not to be used)	[This document]
1	Label TLV	[This document]
2	Label Index TLV	[This document]
3	SRv6 SID TLV	[This document]
4-255	Unassigned	

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.TBD

11. References

11.1. Normative References

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

Dawra, G., Filsfils, C., Talaulikar, K., Raszuk, R., Decraene, B., Zhuang, S., and J. Rabadan, "SRv6 BGP based Overlay Services", draft-ietf-bess-srv6-services-15 (work in progress), March 2022.

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

Patel, K., Raszuk, R., Djernaes, M., Dong, J., and M. Chen, "IPv6 Extensions for Route Target Distribution", draft-ietf-idr-bgp-ipv6-rt-constrain-12 (work in progress), April 2018.

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

Patel, K., Velde, 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-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-19 (work in progress), April 2022.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-22 (work in progress), March 2022.
- [I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Garvia, P. C., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", draft-ietf-spring-srv6-network-programming-28 (work in progress), December 2020.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <<https://www.rfc-editor.org/info/rfc4360>>.
- [RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <<https://www.rfc-editor.org/info/rfc4684>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC5512] Mohapatra, P. and E. Rosen, "The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute", RFC 5512, DOI 10.17487/RFC5512, April 2009, <<https://www.rfc-editor.org/info/rfc5512>>.
- [RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <<https://www.rfc-editor.org/info/rfc5701>>.

- [RFC7311] Mohapatra, P., Fernando, R., Rosen, E., and J. Uttaro, "The Accumulated IGP Metric Attribute for BGP", RFC 7311, DOI 10.17487/RFC7311, August 2014, <<https://www.rfc-editor.org/info/rfc7311>>.
- [RFC7606] Chen, E., Ed., Scudder, J., Ed., Mohapatra, P., and K. Patel, "Revised Error Handling for BGP UPDATE Messages", RFC 7606, DOI 10.17487/RFC7606, August 2015, <<https://www.rfc-editor.org/info/rfc7606>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <<https://www.rfc-editor.org/info/rfc8669>>.

11.2. Informative References

- [I-D.ietf-mpls-seamless-mpls] Leymann, N., Decraene, B., Filsfils, C., Konstantynowicz, M., and D. Steinberg, "Seamless MPLS Architecture", draft-ietf-mpls-seamless-mpls-07 (work in progress), June 2014.
- [RFC3906] Shen, N. and H. Smit, "Calculating Interior Gateway Protocol (IGP) Routes Over Traffic Engineering Tunnels", RFC 3906, DOI 10.17487/RFC3906, October 2004, <<https://www.rfc-editor.org/info/rfc3906>>.

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <<https://www.rfc-editor.org/info/rfc4272>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC6952] Jethanandani, M., Patel, K., and L. Zheng, "Analysis of BGP, LDP, PCEP, and MSDP Issues According to the Keying and Authentication for Routing Protocols (KARP) Design Guide", RFC 6952, DOI 10.17487/RFC6952, May 2013, <<https://www.rfc-editor.org/info/rfc6952>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.

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

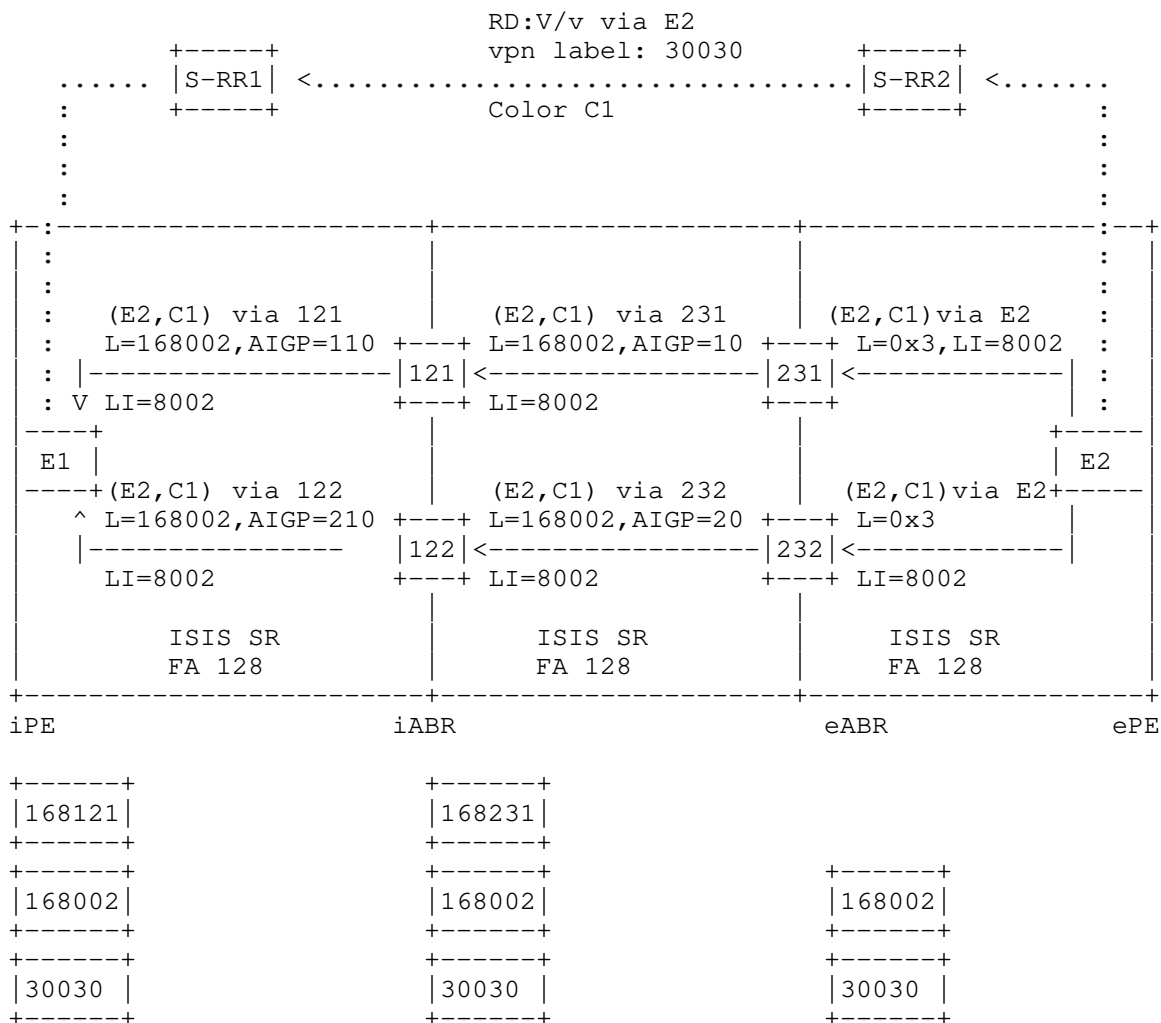


Figure 6: BGP FA Aware transport CAR path

Use case: Provide end to end intent for service flows.

o 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

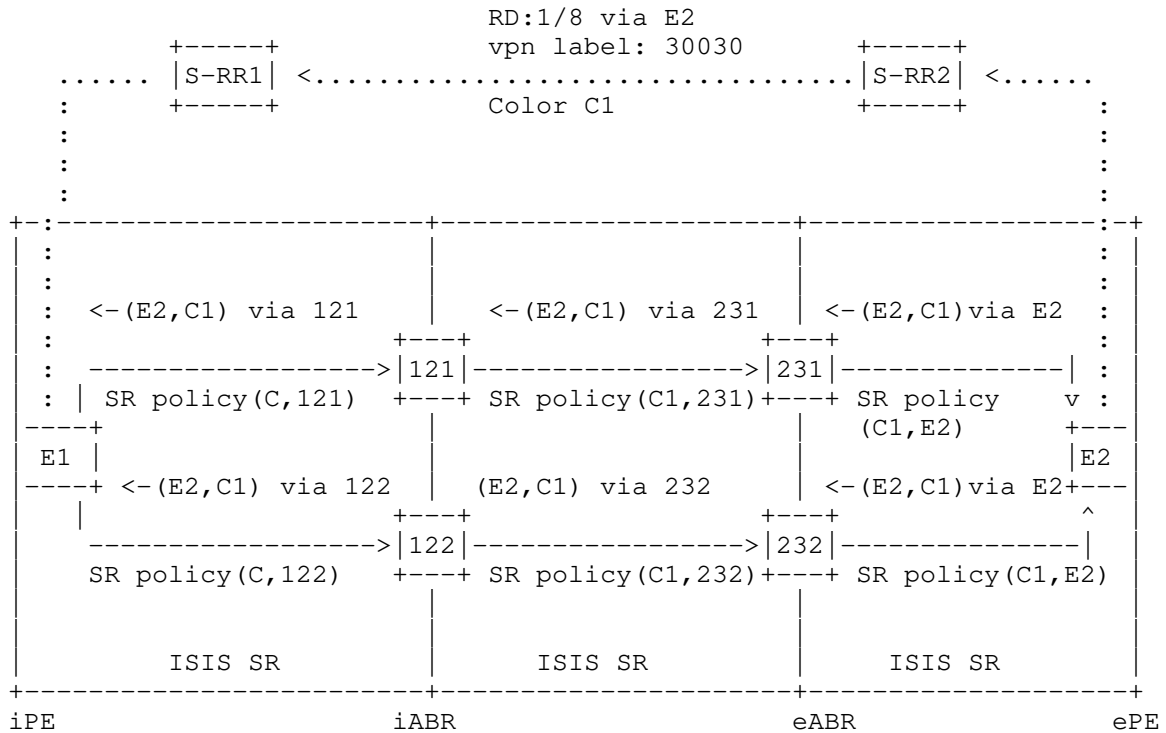


Figure 7: BGP SR policy Aware transport CAR path

Use case: Provide end to end intent for service flows

o 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).
 - o 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

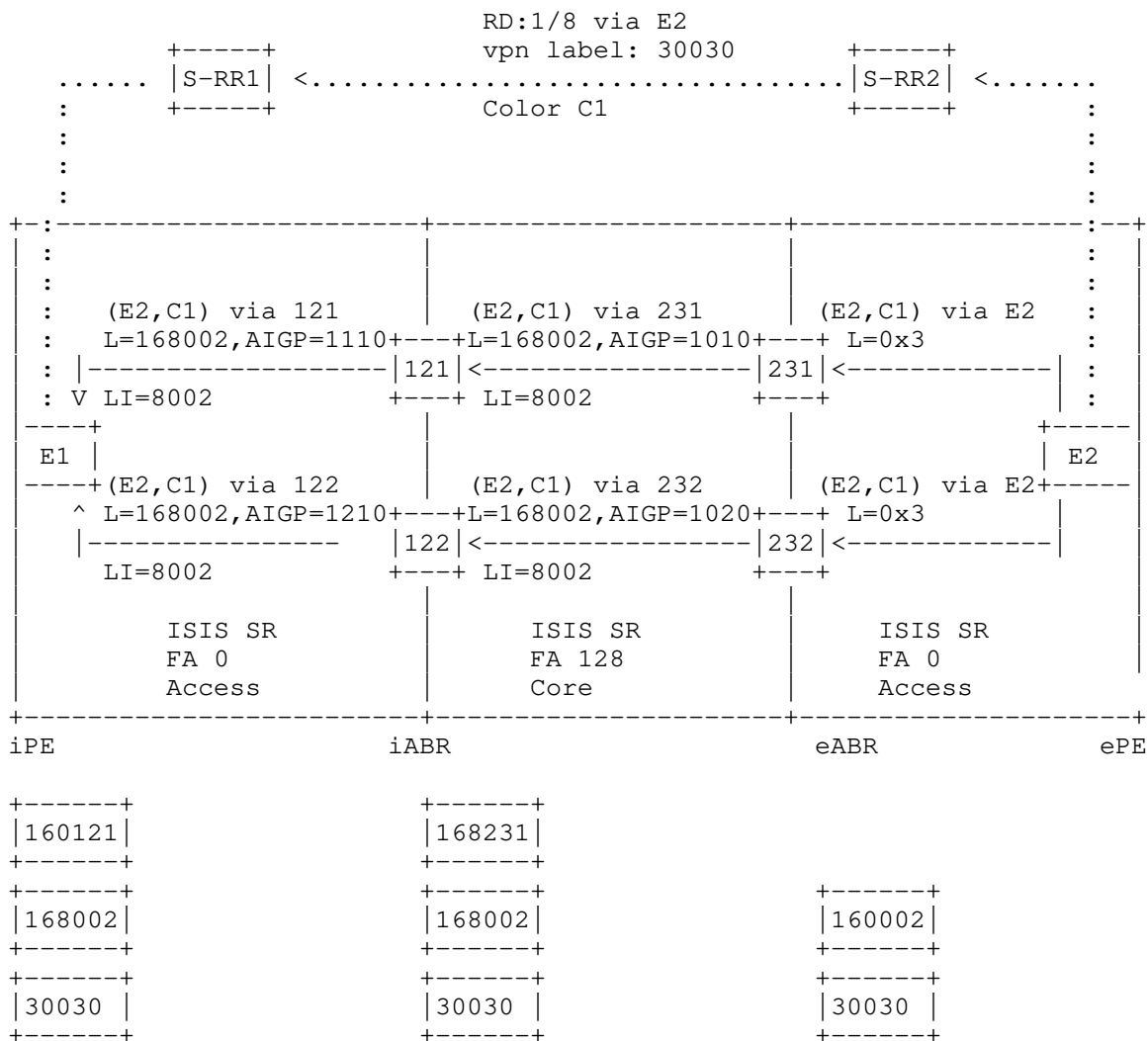


Figure 8: BGP Hybrid FA Aware transport CAR path

Use case: Provide intent for service flows only in Core domain.

o 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

(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 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)
- o 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

- o 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.

- o Reference topology:

```
E1 --- BR1 --- BR2 ..... BR3 ---- BR4 --- E2
    Ci           <----LU---->           Ci
```

- * 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
- o BR1 and BR4 form an over-the-top peering (via RRs as needed) to exchange BGP CAR routes
- o 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
- o BR1 recursively resolves the BGP CAR next-hop for CAR routes learnt from BR4 via the BGP-LU path to BR4
- o BR1 signals the transport discontinuity to E1 via the AIGP TLV, so that E1 can prefer other paths if available
- o BR4 does the same in the reverse direction
- o 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
- o 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

- o 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.
- o The N intents are both signaled end-to-end via BGP CAR routes; as well as realized in the data plane.
- o Appendix A.1 is an example of this case.

B.2. Single color domain containing network domains with N:M color distribution

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

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

- o The original intent in the BGP CAR route is preserved; i.e. route is (E, C1)
- o A BR of D1 attaches LCM-EC with value C1 when advertising to a BR in D2
- o A BR in D2 receiving (E, C1) maps C1 in received LCM-EC to local color, say C2
- o 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.
- o 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
- o On an ingress PE in D2, V/v will resolve via C2
- o 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

Clarence Filsfils
Cisco Systems
Belgium

Email: cfilsfil@cisco.com

Dirk Steinberg
Lapishills Consulting Limited
Germany

Email: dirk@lapishills.com

Luay Jalil
Verizon
USA

Email: luay.jalil@verizon.com

Yuanchao Su
Alibaba, Inc

Email: yitai.syc@alibaba-inc.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Jim Guichard
Futurewei
USA

Email: james.n.guichard@futurewei.com

Ketan Talaulikar
Arrcus, Inc
India

Email: ketant.ietf@gmail.com

Keyur Patel
Arrcus, Inc
USA

Email: keyur@arrcus.com

Haibo Wang
Huawei Technologies
China

Email: rainsword.wang@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 8 October 2022

H. Bidgoli, Ed.
Nokia
D. Voyer
Bell Canada
A. Stone
Nokia
R. Parekh
S. Krier
S. Agrewal
Cisco System, Inc.
6 April 2022

Advertising p2mp policies in BGP
draft-hb-idr-sr-p2mp-policy-05

Abstract

SR P2MP policies are set of policies that enable architecture for P2MP service delivery.

A P2MP policy consists of candidate paths that connects the Root of the Tree to a set of Leaves. The P2MP policy is composed of replication segments. A replication segment is a forwarding instruction for a candidate path which is downloaded to the Root, transit nodes and the leaves.

This document specifies a new BGP SAFI with a new NLRI in order to advertise P2MP policy from a controller to a set of nodes.

This document introduces three new route types within this NLRI, one for P2MP policy and its candidate paths that need to be programmed on the Root node, one for the replication segment incoming SID which uniquely will identify the cross connect and another for each outgoing interface that the packets get replicated to. The last two route types are forwarding instructions that needs to be programmed on the Root, and optionally on Transit and Leaf nodes.

It should be noted that this document does not specify how the Root and the Leaves are discovered on the controller, it only describes how the P2MP Policy and Replication Segments are programmed from the controller to the nodes.

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 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	3
2. Conventions used in this document	4
3. P2MP Policy and Replication Segment Encoding	4
3.1. P2MP Policy SAFI and NLRI	4
3.1.1. P2MP Policy Route - Route Type TBD1	5
3.1.2. Replication segment Route Binding SID- Route type TBD 2	6
3.1.3. Replication segment Route OIF- Route type TBD 3	8
3.2. Tunnel Encapsulation Attribute	9
3.2.1. SR P2MP policy encoding	9
3.2.2. Replication segment Binding SID encoding	10
3.2.3. Replication segment OIF encoding	10
3.3. P2MP Policy Sub-TLVs	11
3.3.1. preference Sub-TLV	11
3.3.2. leaf-list Sub-TLV	11
3.3.3. path-instance Sub-TLV	12
3.3.3.1. active instance-id Sub-TLV	12
3.3.3.2. instance-id Sub-TLV	13
3.4. Replication segment Sub-TLVs	13
3.4.1. Segment list Sub-TLV	14

3.4.2.	Weight sub-tlv	14
3.4.3.	Protection sub-tlv	14
3.4.4.	Segment Sub-TLV	15
4.	P2MP Policy Operation	15
4.1.	Configuration and advertisement of P2MP Policies	16
4.2.	Reception of an P2MP Policy NLRI	16
4.3.	Global Optimization for P2MP LSPs	16
5.	IANA Consideration	17
6.	Security Considerations	17
7.	Acknowledgments	17
8.	References	17
8.1.	Normative References	17
8.2.	Informative References	17
	Authors' Addresses	18

1. Introduction

The draft [draft-ietf-pim-sr-p2mp-policy] defines a variant of the SR Policy [draft-ietf-spring-segment-routing-policy] for constructing a P2MP segment to support multicast service delivery.

A Point-to-Multipoint (P2MP) Policy contains a set of candidate paths and identifies a Root node and a set of Leaf nodes in a Segment Routing Domain. The draft also defines a Replication segment, which corresponds to the state of a P2MP segment on a particular node. The Replication segment is the forwarding instruction for a P2MP LSP at the Root, Transit and Leaf nodes.

For a P2MP segment, a controller may be used to compute a tree from a Root node to a set of Leaf nodes, optionally via a set of replication nodes. A packet is replicated at the root node and optionally on Replication nodes towards each Leaf node.

We define two types of a P2MP segment: Ingress Replication (aka Spray) and Downstream Replication (aka TreeSID).

A Point-to-Multipoint service delivery could be via Ingress Replication (aka Spray in some SR context), i.e., the root unicasts individual copies of traffic to each leaf. The corresponding P2MP segment consists of replication segments only for the root and the leaves.

A Point-to-Multipoint service delivery could also be via Downstream Replication (aka TreeSID in some SR context), i.e., the root and some downstream replication nodes replicate the traffic along the way as it traverses closer to the leaves.

It should be noted that two replication nodes can be connected directly, or they can be connected via unicast SR segment or a segment list.

The leaves and the root of a p2mp policy can be discovered via the multicast protocols or procedures like NG-MVPN [RFC6513] or manually configured on the PCC (CLI) or the PCE.

Based on the discovered root and leaves, the controller builds a P2MP policy and advertise it to the head-end router (i.e. the root of the P2MP Tree). The advertisement uses BGP extensions defined in this document. The controller also calculates the tree path and builds the replication segments on each segment of the tree, Root, Transit and Leaf nodes and downloads the forwarding instructions to the nodes via BGP extensions defined in this document.

SR p2mp policy is a variant of the SR policy and as such it reuses the concept of a candidate path. This draft reuses some of the concepts and TLVs mentioned in [draft-ietf-idr-segment-routing-te-policy]

A candidate path within the P2MP policy can contain multiple path-instances. A path-instance can be viewed as a P2MP LSP. For candidate path global optimization purposes, two or more path-instances can be used to execute make before break procedures.

Each path-instance is a P2MP LSP as such each path-instance needs a set of replication segments to construct its forwarding instructions.

2. Conventions used in this document

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

3. P2MP Policy and Replication Segment Encoding

3.1. P2MP Policy SAFI and NLRI

This document defines a new BGP NLRI, called the P2MP-POLICY NLRI.

A new SAFI is defined: the SR P2MP Policy SAFI, (Codepoint tbd assigned by IANA). The following is the format of the P2MP-POLICY NLRI:

route type	1 octet
length	1 octet
route type specific (variable)	

- * The Route type field defines the encoding of the rest of the P2MP-POLICY NLRI.
- * The length field indicates the length in octets of the route type specific data, excluding route type and length
- * This document defines the following route types:
 - P2MP Policy route: TBD1, this is the actually P2MP policy on the root which contains the candidate paths, its preference and path instances.
 - Replication Segment Binding SID: TBD2, this is part of the replication segment and it is used for programming the incoming SID used to identify a P2MP cross connect.
 - Replication Segment OIF: TBD3, this is a single Outgoing Interface for the P2MP cross connect. It also contains the outgoing SID.

The NLRI containing the SR P2MP Policy is carried in a BGP UPDATE message [RFC4271] using BGP multiprotocol extensions [RFC4760] with an AFI of 1 or 2 (IPv4 or IPv6) and with a SAFI of "TBD" (assigned by IANA from the "Subsequent Address Family Identifiers (SAFI) Parameters" registry).

All other recommendations of [draft-ietf-idr-segment-routing-te-policy] section SR Policy SAFI and NLRI, should be taken into account for P2MP policy.

3.1.1. P2MP Policy Route - Route Type TBD1

Root-ID Length	1 octets
Root-ID	4 or 16 octets (ipv4/ipv6)
Tree-ID	4 octets
Distinguisher	4 octets

- * Root-ID: IPv4/IPv6 address of the head-end (Root) of the p2mp tree, based on AFI.
- * Tree-ID: a unique 4 octets identifier of the p2mp tree on the head- end (root)router.
- * Distinguisher: 4-octets value uniquely identifying the policy in the context of <Tree-ID, Originating Router's IP> tuple. The distinguisher has no semantic value and is solely used by the SR P2MP Policy originator to make unique (from an NLRI perspective) multiple occurrences of the same SR P2MP Policy.

3.1.2. Replication segment Route Binding SID- Route type TBD 2

There can be two type of replication segment, shared and non-shared. A shared replication segment can carry multiple MVPN services or it can be used for Facility Fast reroute protecting multiple P2MP trees. A non-shared tree is used when the label field of the PMSI Tunnel Attribute (PTA) is set to 0 as per [draft-ietf-bess-mvpn-evpn-sr-p2mp]. The Binding SID route type Programs the incoming replication SID on the replication node. Since a replication cross connect has a single incoming replication SID with a set of Outgoing Interfaces, this route type can be used to download the replication SID once for the cross connect.

	Root-ID Length		1 octets
~	Root-ID	~	4 or 16 octets (ipv4/ipv6)
	Tree-ID		4 octets
	Distinguisher		4 octets
	instance-ID		2 octets
	Node-ID Length		1 octets
~	Node-ID	~	4 or 16 octets
	Replication SID Length		1 octets
~	Replication SID	~	4 or 16 octets

- * Root-ID: IPv4/IPv6 address of the head-end (Root) of the p2mp tree based on AFI.
- * Tree-ID: a unique 4 octets identifier of the p2mp tree on the head- end router (Root)
- * instance-id, identifies the path-instance with in the p2mp-policy. Each candidate path can have one, two or more path-instance. Path-instance is used for global optimization of the candidate path via make before break procedures. Instance-ID can be used
- * Distinguisher: 4-octets value uniquely identifying the policy in the context of <Root-ID, Tree-ID> tuple. The distinguisher has no semantic value and is solely used by the SR P2MP Policy originator to make unique (from an NLRI perspective) multiple occurrences of the same SR P2MP Policy.
- * Node-ID: This Node's IPv4/IPv6 address
- * Replication SID: the incoming replication SID used to identify this replication point (MPLS or SRv6). Note the replication SID is not part of the NLRI key.

3.1.3. Replication segment Route OIF- Route type TBD 3

This route type is used to identify and program each out going interface individually for a replication cross connect. Downloading each OIF individually ensures easier modification and programming and will keep the programming of each OIF in par with [draft-ietf-idr-segment-routing-te-policy] . Note: this route type can be used for shared and non-shared replication segment as it was explained in previous sections.

Root-ID Length	1 octets
Root-ID	4 or 16 octets (ipv4/ipv6)
Tree-ID	4 octets
Distinguisher	4 octets
instance-ID	2 octets
Node-ID Length	1 octets
Node-ID	4 or 16 octets
Downstream-Node Length	1 octets
Downstream-Node	4 or 16 octets
Outgoing-TreeSID Length	1 octets
Outgoing-TreeSID	4 or 16 octets

- * Root-ID: IPv4/IPv6 address of the head-end (Root) of the p2mp tree based on AFI.
- * Tree-ID: a unique 4 octets identifier of the p2mp tree on the head- end router (Root)
- * instance-id, identifies the path-instance with in the p2mp-policy. Each candidate path can have one, two or more path-instance. Path-instance is used for global optimization of the candidate path via make before break procedures. Instance-ID can be used

- * Distinguisher: 4-octets value uniquely identifying the policy in the context of <Root-ID, Tree-ID> tuple. The distinguisher has no semantic value and is solely used by the SR P2MP Policy originator to make unique (from an NLRI perspective) multiple occurrences of the same SR P2MP Policy.
- * Node-ID: Node's IPv4/IPv6 address
- * Downstream Node: Downstream Node Identifier
- * Outgoing TreeSID: The outgoing SID for this branch (MPLS or SRv6). Note the outgoing-TreeSID is not part of the NLRI Key.

3.2. Tunnel Encapsulation Attribute

The content of this new NLRI is encoded in the tunnel Encapsulation Attribute originally defined in [RFC9012] using two new Tunnel-Type TLV (codepoint is TBD, assigned by IANA from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry) one for P2MP Policy and another for Replication segment.

3.2.1. SR P2MP policy encoding

SR P2MP Policy SAFI NLRI: <route-type p2mp-policy>

Attributes:

```
Tunnel Encaps Attribute (23)
  Tunnel Type: (TBD, P2MP-Policy)
    Preference
    Policy Name
    Policy Candidate Path Name
    leaf-list (optional)
      remote-end point
      remote-end point
      ...
    path-instance
      active-instance-id
      instance-id
      instance-id
      ...
```

- * Relevant only at the Root.
- * SR P2MP-POLICY NLRI and P2MP Policy route type.
- * Tunnel Encapsulation Attribute is defined in [RFC9012]

- * Tunnel-Type is set to P2MP-Policy Tunnel-Type TBD (assigned by IANA from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry).
- * Policy Name, Policy Candidate Path Name are defined in [draft-ietf-idr-segment-routing-te-policy]
- * Preference, leaf-list, remote-end point and path- instance, instance-ids are defined in this document.
- * Additional sub-TLVs may be defined in the future.

3.2.2. Replication segment Binding SID encoding

```
replication segment Binding SID SAFI NLRI:
    <route-type non-sahred/shared
      tree replication-segment-binding-sid>
```

This route type has no additional sub-TLVs, and it is only meant to download the incoming SID for the replication cross connect.

3.2.3. Replication segment OIF encoding

```
replication segment SAFI NLRI: <route-type non-sahred/shared
                                tree replication-segment-oif>
```

Attributes:

```
Tunnel Encaps Attribute (23)
  Tunnel Type: (TBD Replication-Segment-oif)
    segment-list
      weight (optional)
      protection (optional, must be present when protection flag is enabled
for downstream-nodes)
      segment
      segment
      ...
    segment-list
      weight (optional)
      protection (optional, must be present when protection flag is enabled
for downstream-nodes)
      segment
      segment
      ...
    segment-list (protection segment list)
      protection (protecting the first segment list, can't have weight sub-
tlv)
      segment
      segment
      ...
    ...
  ...
```

- * SR P2MP-POLICY NLRI and non-shared tree Replication segment route type or shared tree Replication segment route type.
- * Tunnel Encapsulation Attribute is defined in [RFC9012].
- * Tunnel-Type is set to Replication Segment OIF Tunnel Type, TBD (assigned by IANA from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry).
- * segment-list are defined in this document.
- * Additional sub-TLVs may be defined in the future.

3.3. P2MP Policy Sub-TLVs

EACH P2MP policy NLRI represents a candidate path for a P2MP policy. A P2MP policy can have multiple candidate paths and would need multiple P2MP policy NLRI to download all the candidate paths.

3.3.1. preference Sub-TLV

As defined in preference Sub-TLV section in [draft-ietf-idr-segment-routing-te-policy] the candidate path with highest preference is the active candidate path.

3.3.2. leaf-list Sub-TLV

The leaf list sub-tlv identifies a set of leaves for the tree. Each leaf is a remote endpoint as defined in [RFC9012] The leaf-list sub-tlv is optional. The PCE can choose to download the leaf list every time it is configured or learns a new leaf. If the PCE chooses to download this optional sub-tlv it should download the entire set of the end-points every time the endpoint list has been modified. The leaf list has informational value only hence why it is optional and it is not required for the root PE to operate. However, it must be noted that in some cases the end-points list can become very large with 100s of leaves.

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	2	3	4	5	6	7	8	9
Type										Length										RESERVED																			
// sub-TLVs //																																							

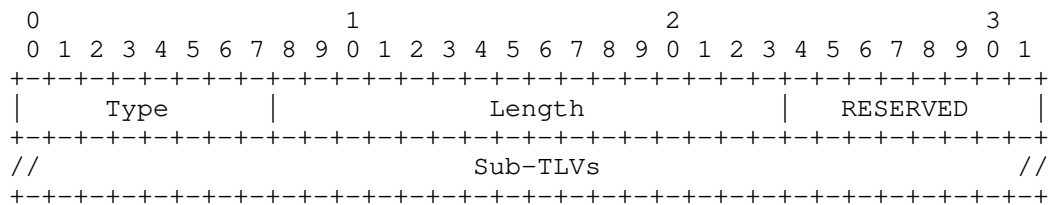
- * Type: TBD, 1 octet

- * Length: 2 octets, the total length (not including the Type and Length fields) of the sub-TLVs encoded within the leaf-list sub-TLV.
- * RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- * sub-TLVs: One or more remote endpoint sub-TLVs. Note the remote endpoint object is defined in [RFC9012]

3.3.3. path-instance Sub-TLV

The path instance sub-tlv contains a set of instance-ids (P2MP LSPs). These LSPs can be used for MBB procedure under a candidate path. Each LSP Instance-id has a unique id (4 octets) with in the <root node, P2MP policy>, in other word it is unique per <root node, tree-id>. The PCE SHOULD always download all instance-ids to the node. The active instance is identified via the active instance-id sub-tlv.

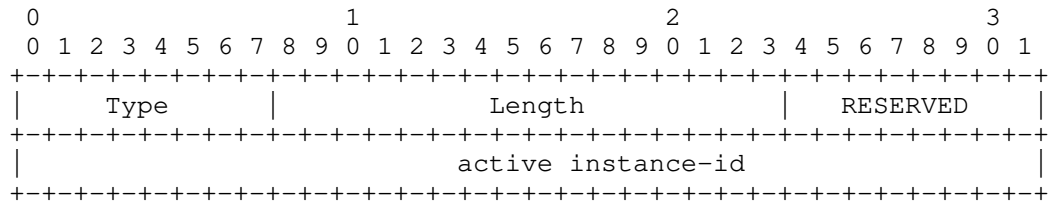
The P2MP LSP and its replication segments should be configured from root to the leaves first before the PCE switches that active instance-id to this new instance.



- * Type: TBD, 1 octet
- * Length: 2 octets, the total length (not including the Type and Length fields) of the sub-TLVs encoded within the Segment List sub-TLV.
- * RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt
- * sub-TLVs: * active instance-id * one or more instance-id

3.3.3.1. active instance-id Sub-TLV

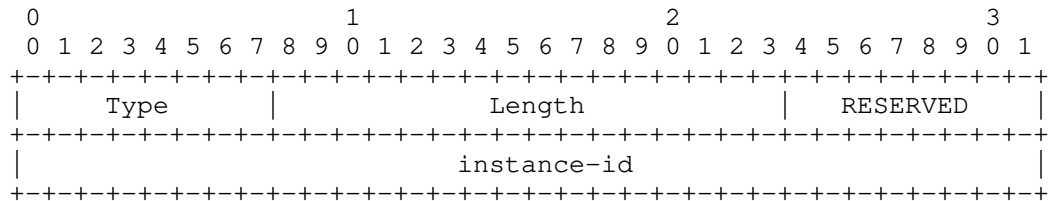
The Active instance-id is used to identify the P2MP LSP which should be active amongst the collection of instances.



- * Type: TBD.
- * Length: the total length (not including the Type and Length fields) of the sub-TLVs encoded within the Segment List sub-TLV.
- * RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- * active instant-id: The identifier of the active instance-id

3.3.3.2. instance-id Sub-TLV

Multiple Instance-ids can be programmed for a candidate path.



- * Type: TBD
- * Length: the total length (not including the Type and Length fields) of the sub-TLVs encoded within the Segment List sub-TLV.
- * RESERVED: 1 octet of reserved bits. SHOULD be unset on transmission and MUST be ignored on receipt.
- * instan-id: a 32 bit unique identifier. The instance-id is unique with in the context of the <root node, p2mp policy>

3.4. Replication segment Sub-TLVs

3.4.1. Segment list Sub-TLV

The segment list Sub-TLV is defined in [draft-ietf-spring-segment-routing-policy]. The segment-list Sub-TLV contains one or more segment Sub-TLVs. Two replication segments can be directly connected via a replication sid or can be connected via a unicast segment list and a replication sid. In the later case the replication sid needs to be at the bottom of the unicast segment list.

3.4.2. Weight sub-tlv

The Weight sub-TLV is optional and is as defined in [draft-ietf-idr-segment-routing-te-policy]. With in the downstream node sub-tlv, there can be one or more segment list used for ECMP. In this case the weight sub-tlv can provide weighted ECMP.

3.4.3. Protection sub-tlv

Protection sub-tlv is optional, if FRR is desired for the downstream node this sub-tlv can be used to identify the protection segment list. To identify protection segment list this sub-tlv provides a segment list identifier. If protection is desired under the endpoint all the segment lists should have this sub-tlv. A protection segment list can not have a weight sub-tlv and it can not participate in ECMP. That said a segment list that is being protected can have a weight sub-tlv and participate in ECMP.

In general protection segment list is used only if replication segments are directly connected and there is no unicast segment list connecting two replication segment. If there is a unicast replication segment connecting the two replication sid, then the unicast protection mechanism can be exercise and there is no need for this protection sub-tlv, hence why this sub-tlv is optional.

0								1								2								3								
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	
Type								Length								Flags								P	RESERVED							
segment list id																protection segment list id																

* Type : tbd, 1 octet.

* Length: 8

- * Flag: 1 octet, the P bit is set when this segment list is protected by another segment list for the downstream node
- * segment list id: the segment list id
- * protection segment list id: the segment list id that is being used as protection.

3.4.4. Segment Sub-TLV

The segment sub-Tlv is identified in [draft-ietf-idr-segment-routing-te-policy]. As it was mentioned before two replication segments can be connected directly to each other or via a segment list. If they are connected directly to each other then the segment list can be constructed via:

- * If the replication segment is steered via IPv4 or IPv6 nexthops or interface then the segment type E or G can be used with the new R flag set.
- * If the replication segment is steered via a SR Unicast node or adjacency SID then segment type A can be used with the new R flag set. Unicast SR segment types can also be configured for steering.

If they are connected via SR domain then the segment list can contain multiple different types of SIDs, such as Node, Adjacency or Binding SIDs. In this case the replication sid is at the bottom of the stack and of type A with the R flag set. The SR node/adjacency or binding sids steer the packet through a SR domain until it reaches another replication segment. where the bottom of the stack replication sid identifies the forwarding information on that replication segment.

It should be noted that the segment sub-TLV is only used to program the unicast SR Segment or outgoing interface for the replication SID outgoing interface. The outgoing tree SID it self is programmed in the appropriate route type.

4. P2MP Policy Operation

Inline with [draft-ietf-idr-segment-routing-te-policy] the consumer of an P2MP Policy is not the BGP process. The BGP process is used for distributing the P2MP policy NLRI and its route-types but its installation and use is outside the scope of BGP. The detail for P2MP Policy can be found in [draft-ietf-pim-sr-p2mp-policy]

4.1. Configuration and advertisement of P2MP Policies

The controller usually is connected to the receivers via a route reflector. As such one or more route-target SHOULD be attached to the advertisement of P2MP Policy NLRI and its route-type. Each route target identifies one head-end (root nodes) for P2MP Policy route or one or more head-end, transit and leaf nodes for the Non- Shared/ Shared Tree Replication Segment route, for the advertised P2MP Policy.

4.2. Reception of an P2MP Policy NLRI

When a BGP speaker receives an P2MP Policy NLRI the following rules apply:

- * The P2MP Policy update MUST have either the NO_ADVERTISE community or at least one route-target extended community in IPv4-address format. If a router supporting this document receives an P2MP Policy update with no route-target extended communities and no NO_ADVERTISE community, the update MUST NOT be processed. Furthermore, it SHOULD be considered to be malformed, and the "treat-as-withdraw" strategy of [RFC7606] is applied.
- * If one or more route-targets are present, then at least one route-target MUST match one of the BGP Identifiers of the receiver in order for the update to be considered usable. The BGP Identifier is defined in [RFC4271] as a 4 octet IPv4 address. Therefore the route- target extended community MUST be of the same format.
- * If one or more route-targets are present and no one matches any of the local BGP Identifiers, then, while the P2MP Policy NLRI is acceptable, it is not usable on the receiver node.

4.3. Global Optimization for P2MP LSPs

When a P2MP LSP needs to be optimized for any reason (i.e. it is taking on an FRR Path or new routers are added to the network) a global optimization is possible. Note that optimization works per candidate path. Each candidate path is capable of global optimization. To do so each candidate path contains two or more path- instances. Each path instance is a P2MP LSP, each P2MP LSP is identified via a path-instance-id (equivalent to an lsp-id [RFC3209]). After calculating an optimized P2MP LSP path the PCE will program the candidate path with a 2nd path instance and its set of replication segments for this path-instance on the root, transit and leaf nodes. After the optimized LSP replication segments are downloaded a MBB procedure is performed and the previous instance of the path instance is deleted and removed from head-end node and its

corresponding replication segments from head-end, transit and leaves.

5. IANA Consideration

- * A new SAFI is defined: the SR P2MP Policy SAFI, (Codepoint tbd assigned by IANA)
- * 3 new Route type field defines the encoding of the rest of the P2MP- POLICY SAFI
 - P2MP Policy Route
 - Replication Segment Binding Sid
 - Replication Segment OIF
- * Two new Tunnel type to be assigned by IANA
 - P2MP-Policy Tunnel-Type
 - Replication Segment OIF Tunnel Type

6. Security Considerations

TBD

7. Acknowledgments

8. References

8.1. Normative References

[RFC2119] "S. Bradner "Key Words for use in RFCs to Indicate Requirement levels"", October 2019.

8.2. Informative References

[draft-ietf-bess-mvpn-evpn-sr-p2mp]
"R. Parekh, C. Filsfils, A.V. Venkateswaran, H. Bidgoli, D. Voyer, Z. Zhang "Multicast and Ethernet VPN with Segment Routing P2MP"".

[draft-ietf-idr-segment-routing-te-policy]
"s. Previdi, C. Filsfils, K. Talaulikar, P. Mattes, D. Jain, S. Lin "Advertise Segment Routing Policies in BGP"".

- [draft-ietf-pim-sr-p2mp-policy]
"D. Voyer, C. Filsfils, R.Prekh, H.bidgoli, Z. Zhang,
"draft-ietf-pim-sr-p2mp-policy"", October 2019.
- [draft-ietf-spring-segment-routing-policy]
"C. Filsfils, K. Talaulikar, D. Voyer, A. Bogdanov, P.
Mattes "Segment Routing Policy Architecture".
- [RFC4271] "Y. Rekhter, T. Li, S. Hares "A Border Gateway Protocol 4
(BGP-4) "".
- [RFC4760] "T. Bates, R. Chandra, D. Katz, Y. Rekhter "Multiprotocol
Extensions for BGP-4"".
- [RFC6513] "E. Rosen, R. Aggarwal "Multicast in MPLS/BGP IP VPNs"".
- [RFC7606] "e. Chen, J. Scudder, P. Mohapatra, K. Patel "Revised
Error handling for BGP UPDATE Messages"".
- [RFC9012] "K. Patel, G. Van de Velde, S. Sangli, J. Scudder "The BGP
Tunnel Encapsulation Attribute"".

Authors' Addresses

Hooman Bidgoli (editor)
Nokia
Ottawa
Canada
Email: hooman.bidgoli@nokia.com

Daniel Voyer
Bell Canada
Montreal
Canada
Email: daniel.yover@bell.ca

Andrew Stone
Nokia
Ottawa
Canada
Email: andrew.stone@nokia.com

Rishabh Parekh
Cisco System, Inc.
San Jose,
United States of America
Email: riparekh@cisco.com

Serge Krier
Cisco System, Inc.
Rixensart
Belgium
Email: sekrier@cisco.com

Swadesh Agrewal
Cisco System, Inc.
San Jose,
United States of America
Email: swaagraw@cisco.com

Interdomain Routing
Internet-Draft
Intended status: Standards Track
Expires: 7 September 2022

M. Jethanandani
Kloud Services
K. Patel
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 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. 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 AFI 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 liveness 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 NLRI's.
- * 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

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

Table 1

2. Model overview

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

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



```

|   +---n backward-transition
|   |   ...
|   +---x clear {bt:clear-neighbors}?
|   |   ...
+--rw peer-groups
|   +--rw peer-group* [name]
|   |   ...
+--rw interfaces
|   +--rw interface* [name]
|   |   ...
+--ro rib
|   +--ro attr-sets
|   |   ...
|   +--ro communities
|   |   ...
|   +--ro ext-communities
|   |   ...
|   +--ro large-communities
|   |   ...
|   +--ro afi-safis
|   |   ...

```

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

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

In order to allow common configuration to be applied to a set of neighbors, all neighbor configuration options are available within a peer-group. A neighbor is associated 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:

URI: urn:ietf:params:xml:ns:yang:ietf-bgp
URI: urn:ietf:params:xml:ns:yang:ietf-bgp-policy
URI: urn:ietf:params:xml:ns:yang:ietf-bgp-types

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

5.2. YANG Module Name Registration

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

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

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

```
name: ietf-bgp-types
namespace: urn:ietf:params:xml:ns:yang: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
      (NMDA Version).";
  }
  import ietf-routing-policy {
    prefix rt-pol;
    reference
      "RFC ZZZZ, A YANG Data Model for Routing Policy Management.";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343, A YANG Data Model for Interface Management.";
  }
  import ietf-bgp-types {
    prefix bt;
    reference
      "RFC XXXX, BGP YANG Model for Service Provider Network.";
  }
  import ietf-bfd-types {
    prefix bfd-types;
    reference
      "I-D.ietf-bfd-rfc9127-bis: YANG Data Model for
        Bidirectional Forward Detection (BFD).";
  }
  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-key-chain {
    prefix key-chain;
    reference
      "RFC 8177: YANG Key Chain.";
  }
  import ietf-tcp {
    prefix tcp;
    reference
      "I-D.ietf-tcpm-yang-tcp: Transmission Control Protocol (TCP)
        YANG Model.";
  }
  include ietf-bgp-common {
    revision-date 2022-03-06;
  }
}
```

```
include ietf-bgp-common-multiprotocol {
  revision-date 2022-03-06;
}
include ietf-bgp-common-structure {
  revision-date 2022-03-06;
}
include ietf-bgp-neighbor {
  revision-date 2022-03-06;
}
include ietf-bgp-rib-types {
  revision-date 2022-03-06;
}
include ietf-bgp-rib {
  revision-date 2022-03-06;
}
include ietf-bgp-rib-attributes {
  revision-date 2022-03-06;
}
include ietf-bgp-rib-tables {
  revision-date 2022-03-06;
}
```

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

This model supports the following BGP configuration level hierarchy:

```
BGP
|
```

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

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
    reemote-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
```

```
        "RFC 4273, Definitions of Managed Objects for
        BGP-4.";
    }
    leaf fsm-established-transitions {
        type yang:counter32;
        config false;
        description
            "The total number of times the BGP FSM
            transitioned into the established state
            for this peer.";
        reference
            "RFC 4271, Section 8.";
    }
    container messages {
        config false;
        description
            "Counters for BGP messages sent and received from the
            neighbor";
        leaf in-total-messages {
            type yang:counter32;
            config false;
            description
                "The total number of messages received
                from the remote peer on this connection.";
            reference
                "RFC 4271, Section 4.";
        }
        leaf out-total-messages {
            type yang:counter32;
            config false;
            description
                "The total number of messages transmitted to
                the remote peer on this connection.";
            reference
                "RFC 4271, Section 4.";
        }
    }
    leaf in-update-elapsed-time {
        type yang:gauge32;
        units "seconds";
        config false;
        description
            "Elapsed time (in seconds) since the last BGP
            UPDATE message was received from the peer.
            Each time in-updates is incremented,
            the value of this object is set to zero (0).";
        reference
            "RFC 4271, Section 4.3.
            RFC 4271, Section 8.2.2, Established state.";
```



```
}
container sent {
  description
    "Counters relating to BGP messages sent to the
    neighbor";
  uses bgp-neighbor-counters-message-types-state;
}
container received {
  description
    "Counters for BGP messages received from the
    neighbor";
  uses bgp-neighbor-counters-message-types-state;
}
}
container queues {
  config false;
  description
    "Counters related to queued messages associated with
    the BGP neighbor";
  leaf input {
    type uint32;
    description
      "The 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 off 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.";
    }
}
```



```
}

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 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.";
}

grouping neighbor-group-timers-config {
  description
```

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

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

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

      If the configured value of hold-time object was
      a value of (0), then when read this object MUST have a
      value of (0) also.";
    reference
      "RFC 4271, Section 4.2.
      RFC 4271, Section 10.";
  }
  leaf keepalive {
    type uint16 {
      range "0..21845";
    }
    units "seconds";
```

```
description
  "When used as a configuration (rw) value, this Time interval
  (in seconds) for the KeepAlive timer configured for this BGP
  speaker with this peer. A reasonable maximum value for this
  timer would be one-third of the configured hold-time.

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

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

  The actual time interval for the KEEPALIVE messages is
  indicated by operational value of keepalive."
reference
  "RFC 4271, 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.";
  }
}
```

```
        description
            "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 use a single path.";
      }
    }
  }
  container ibgp {
    description
      "Multi-Path parameters for iBGP";
    leaf maximum-paths {
      type uint32;
      default "1";
      description
        "Maximum number of parallel paths to consider when using
        iBGP multi-path. The default is to use a single path";
    }
  }
}

grouping route-selection-options {
  description
    "Configuration and state relating to route selection options";
  container route-selection-options {
    description
      "Parameters relating to options for route selection";
    leaf always-compare-med {
      type boolean;
      default "false";
      description
        "Compare multi-exit discriminator (MED) value from
        different ASes when selecting the best route. The default
        behavior is to only compare MEDs for paths received from
        the same AS.";
    }
  }
}
```

```
leaf ignore-as-path-length {
  type boolean;
  default "false";
  description
    "Ignore the AS path length when selecting the best path.
     The default is to use the AS path length and prefer paths
     with a shorter length.";
}
leaf external-compare-router-id {
  type boolean;
  default "true";
  description
    "When comparing similar routes received from external BGP
     peers, use the router-id as a criterion to select the
     active path.";
}
leaf advertise-inactive-routes {
  type boolean;
  default "false";
  description
    "Advertise inactive routes to external peers. The default
     is to only advertise active routes.";
  reference
    "I-D.ietf-idr-best-external: Advertisement of the best
     external route in BGP.";
}
leaf enable-aigp {
  type boolean;
  default "false";
  description
    "Flag to enable sending / receiving accumulated IGP
     attribute in routing updates";
  reference
    "RFC 7311: AIGP Metric Attribute for BGP.";
}
leaf ignore-next-hop-igp-metric {
  type boolean;
  default "false";
  description
    "Ignore the IGP metric to the next-hop when calculating BGP
     best-path. The default is to select the route for which
     the metric to the next-hop is lowest";
}
leaf enable-med {
  type boolean;
  default "false";
  description
    "Flag to enable sending/receiving of MED metric attribute
```

```
        in routing updates.";
    }
    container med-plus-igp {
        leaf enabled {
            type boolean;
            default "false";
            description
                "When enabled allows BGP to use MED and IGP values
                 defined below to determine the optimal route.";
            reference
                "RFC 4451: BGP MED Considerations.";
        }
        leaf igp-multiplier {
            type uint16;
            default 1;
            description
                "Specifies an IGP cost multiplier.";
            reference
                "RFC 4451: BGP MED Considerations.";
        }
        leaf med-multiplier {
            type uint16;
            default 1;
            description
                "Specifies a MED multiplier.";
            reference
                "RFC 4451: BGP MED Considerations.";
        }
        description
            "The med-plus-igp option enables BGP to use the sum of
             MED multiplied by a MED multiplier and IGP cost multiplied
             by IGP cost multiplier to select routes when MED is
             required to determine the optimal route.";
    }
}

grouping state {
    description
        "Grouping containing common counters relating to prefixes and
         paths";
    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 (<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.";
}

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";
      }
      description
        "IPv4 unicast configuration options";
      // include common IPv[46] unicast options
      uses mp-ipv4-ipv6-unicast-common;
      // placeholder for IPv4 unicast specific configuration
    }
    container ipv6-unicast {
      when "../name = 'bt:ipv6-unicast'" {
        description
          "Include this container for IPv6 Unicast specific
          configuration";
      }
      description
        "IPv6 unicast configuration options";
      // include common IPv[46] unicast options
      uses mp-ipv4-ipv6-unicast-common;
      // placeholder for IPv6 unicast specific configuration
      // options
    }
    container ipv4-labeled-unicast {
      when "../name = 'bt:ipv4-labeled-unicast'" {
        description
          "Include this container for IPv4 Labeled Unicast specific
          configuration";
      }
      description
        "IPv4 Labeled Unicast configuration options";
    }
  }
}
```

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

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

// Common groupings across multiple AFI,SAFIs
```



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

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

grouping mp-l3vpn-ipv4-ipv6-unicast-common {
  description
    "Common configuration applied across L3VPN for IPv4
```

```
        and IPv6";
        // placeholder -- specific configuration options that are generic
        // across IPv[46] unicast address families.
        uses mp-all-afi-safi-common;
    }

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

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

    // Config groupings for common groups

    grouping mp-all-afi-safi-common-prefix-limit-config {
        description
            "Configuration parameters relating to prefix-limits for an
            AFI-SAFI";
    }
}
<CODE ENDS>

<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
  (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.";
  }

  grouping structure-neighbor-group-logging-options {
    description
      "Structural grouping used to include error handling
      configuration and state for both BGP neighbors and groups";
    container logging-options {
      description
        "Logging options for events related to the BGP neighbor or
        group";
      leaf log-neighbor-state-changes {
        type boolean;
        default "true";
        description
          "Configure logging of peer state changes. Default is to
          enable logging of peer state changes.

          Note: Documenting 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";
    }
  }
}
```

```
    }  
  }  
  
  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
  (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.";
}

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 thsi 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-SAFI";
    }
    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-SAFI.";
        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-SAFI.";
        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;
}
}
}
<CODE ENDS>

```

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.";
```

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

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

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

  identity ipv6-unicast {
    base afi-safi-type;
    description
      "IPv6 unicast (AFI,SAFI = 2,1)";
    reference
      "RFC4760: Multiprotocol Extensions 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 4364: BGP/MPLS IP Virtual Private Networks (VPNs).";
  }

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

  identity l3vpn-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 0xFFFF0001.";
    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
        0xFFFF0002.";
    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
        0xFFFF0003.";
    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 0xFFFF0004.";
    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 uint32;
    type string {
      pattern '([0-9]|[1-9][0-9]{1,3}|[1-5][0-9]{4})|'
        + '6[0-5][0-9]{3}|66[0-4][0-9]{2})|'
        + '665[0-2][0-9]|6653[0-5]):'
        + '([0-9]|[1-9][0-9]{1,3}|[1-5][0-9]{4})|'
        + '6[0-5][0-9]{3}|66[0-4][0-9]{2})|'
        + '665[0-2][0-9]|6653[0-5])';
    }
  }
  description
    "Type definition for standard community attributes.";
  reference
    "RFC 1997 - BGP Communities Attribute";
}

typedef bgp-ext-community-type {
  type union {
    type string {
      // Type 1: 2-octet global and 4-octet local
      //           (AS number)           (Integer)
      pattern '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{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 {
      // Type 2: 4-octet global and 2-octet local
      //           (ipv4-address)       (integer)
      pattern '(([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-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:'
        + '(([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|'
        + '25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9][0-9]|'
        + '2[0-4][0-9]|25[0-5]):'
        + '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{4}|'
        + '[1-9][0-9]{1,4}|[0-9])';
}
}
description
    "Type 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 {
    // 4-octets global:4-octets local part-1:4-octets local part-2.
    pattern '(4[0-2][0-9][0-4][0-9][0-6][0-7][0-2][0-9][0-6]|'
      + '[1-3][0-9]{9}|[1-9]([0-9]{1,7})?[0-9]|[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]{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]{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";
}
}
<CODE ENDS>
```

7.3. BGP policy data

```
<CODE BEGINS> file "ietf-bgp-policy@2022-03-06.yang"
module ietf-bgp-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-policy";
  prefix bp;

  // import some basic types

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

"WG Web: <<http://datatracker.ietf.org/wg/idr>>
WG List: <idr@ietf.org>

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

description

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

Copyright (c) 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
```

```

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

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

typedef bgp-set-med-type {
    type union {
        type uint32;
        type string {
            pattern '^[+-]([0-9]{1,8}|[0-3][0-9]{1,9}|4[0-1][0-9]{1,8}|
                + '428[0-9]{1,7}|429[0-3][0-9]{1,6}|42948[0-9]{1,5}|
                + '42949[0-5][0-9]{1,4}|429496[0-6][0-9]{1,3}|
                + '4294971[0-9]{1,2}|42949728[0-9]|42949729[0-5]))$';
        }
        type enumeration {
            enum igp {
                description

```

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

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

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

// 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 bt:bgp-std-community-type;
        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 pref 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/"
                }
            }
        }
    }
}
```

```
        + "bgp-defined-sets/"
        + "community-sets/community-set/name";
    }
    description
        "References a defined community set by name";
    }
}

container set-ext-community {
    description
        "Action to set the extended community attributes of the
        route, along with options to modify how the community is
        modified. Extended communities may be set using an inline
        list OR a reference to an existing defined set (but not
        both).";

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

    choice method {
        description
            "Indicates the method used to specify the extended
            communities for the set-ext-community action";
        case inline {
            leaf-list communities {
                type 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.";
      }
    }
  }
}
}

```

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

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
```


"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 dot net).";

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.";

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

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

```
        passed.";
    reference
        "RFC 4456 - BGP Route Reflection: An Alternative to Full
        Mesh Internal BGP (IBGP)";
}
leaf aigp-metric {
    type uint64;
    description
        "BGP path attribute representing the accumulated IGP
        metric for the path";
    reference
        "RFC 7311 - The Accumulated IGP Metric Attribute for BGP";
}
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 {
```

```
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
```

```
    "A grouping for all extended community parameters.";

    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";
    }
}

grouping large-community-attributes {
    description
        "A grouping for all large community parameters.";

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

grouping rib {
    description
        "Grouping for rib.";
    container rib {
        config false;
        uses attr-sets;
        container communities {
            description
                "Enclosing container for the list of community attribute
                sets.";
            list community {
                key "index";
                config false;
                description
                    "List of path attributes that may be in use by multiple
                    routes in the table.";
                leaf index {
                    type uint64;
                    description
                        "System generated index for each attribute set. The
                        index is used to reference an attribute set from a
```

```
        specific path. Multiple paths may reference the same
        attribute set.";
    }
    uses bgp-community-attr-state;
}
container ext-communities {
    description
        "Enclosing container for the list of extended community
        attribute sets.";
    list ext-community {
        key "index";
        config false;
        description
            "List of path attributes that may be in use by multiple
            routes in the table.";
        leaf index {
            type uint64;
            description
                "System generated index for each attribute set. The
                index is used to reference an attribute set from a
                specific path. Multiple paths may reference the same
                attribute set.";
        }
        uses ext-community-attributes;
    }
}
container large-communities {
    description
        "Enclosing container for the list of large community
        attribute sets.";
    list large-community {
        key "index";
        config false;
        description
            "List of path attributes that may be in use by multiple
            routes in the table.";
        leaf index {
            type uint64;
            description
                "System generated index for each attribute set. The
                index is used to reference an attribute set from a
                specific path. Multiple paths may reference the same
                attribute set.";
        }
        uses large-community-attributes;
    }
}
```



```
container afi-safis {
  config false;
  description
    "Enclosing container for address family list.";
  list afi-safi {
    key "name";
    description
      "List of afi-safi types.";
    leaf name {
      type identityref {
        base bt:afi-safi-type;
      }
      description
        "AFI, SAFI name.";
    }
  }
  container ipv4-unicast {
    when "../name = 'bt:ipv4-unicast'" {
      description
        "Include this container for IPv4 unicast RIB.";
    }
    description
      "Routing tables for IPv4 unicast -- active when the
        afi-safi name is ipv4-unicast.";
  }

  container loc-rib {
    config false;
    description
      "Container for the IPv4 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

```

```
    "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.

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 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 ENDS>
```

```
<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
    (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 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 ENDS>

<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
  (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 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/"
        + "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.";
        }
      }
    }
  }
  description
    "Action commands to clear routes governed by a if-feature.";
}

grouping ipv4-adj-rib-common {
  description
    "Common structural grouping for each IPv4 adj-RIB table.";
  container routes {
    config false;
    description
      "Enclosing container for list of routes in the routing
```

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

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

```
        uses rib-ext-route-annotations;
    }
}

grouping ipv6-adj-rib-common {
    description
        "Common structural grouping for each IPv6 adj-RIB table.";
    container routes {
        config false;
        description
            "Enclosing container for list of routes in the routing
            table.";
        list route {
            key "prefix path-id";
            description
                "List of routes in the table.";
            leaf prefix {
                type inet:ipv6-prefix;
                description
                    "Prefix for the route.";
            }
            uses bgp-adj-rib-attr-state;
            uses bgp-adj-rib-common-attr-refs;
            uses bgp-common-route-annotations-state;
            uses bgp-unknown-attr-top;
            uses rib-ext-route-annotations;
        }
    }
}

grouping ipv6-adj-rib-in-post {
    description
        "Common structural grouping for the IPv6 adj-rib-in
        post-policy table.";
    container routes {
        config false;
        description
            "Enclosing container for list of routes in the routing
            table.";
        list route {
            key "prefix path-id";
            description
                "List of routes in the table.";
            leaf prefix {
                type inet:ipv6-prefix;
                description
                    "Prefix for the route.";
            }
        }
    }
}
```

```
    }
    uses bgp-adj-rib-attr-state;
    uses bgp-adj-rib-common-attr-refs;
    uses bgp-common-route-annotations-state;
    uses bgp-adj-rib-in-post-route-annotations-state;
    uses bgp-unknown-attr-top;
    uses rib-ext-route-annotations;
  }
}
}
}
<CODE ENDS>
```

8. Contributors

Previous versions of this document saw contributions from Anees Shaikh, Rob Shakir, Kevin D'Souza, Alexander Clemm, Aleksandr Zhadkin, and 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-Nygaard, Adam Simpson, Puneet Sood, Jason Sterne, Jeff Tantsura, Jim Uttaro, and Gunter Vandeveld.

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

- [RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", RFC 1997, DOI 10.17487/RFC1997, August 1996, <<https://www.rfc-editor.org/info/rfc1997>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC2439] Villamizar, C., Chandra, R., and R. Govindan, "BGP Route Flap Damping", RFC 2439, DOI 10.17487/RFC2439, November 1998, <<https://www.rfc-editor.org/info/rfc2439>>.
- [RFC2918] Chen, E., "Route Refresh Capability for BGP-4", RFC 2918, DOI 10.17487/RFC2918, September 2000, <<https://www.rfc-editor.org/info/rfc2918>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4451] McPherson, D. and V. Gill, "BGP MULTI_EXIT_DISC (MED) Considerations", RFC 4451, DOI 10.17487/RFC4451, March 2006, <<https://www.rfc-editor.org/info/rfc4451>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.
- [RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", RFC 4659, DOI 10.17487/RFC4659, September 2006, <<https://www.rfc-editor.org/info/rfc4659>>.
- [RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", RFC 4724, DOI 10.17487/RFC4724, January 2007, <<https://www.rfc-editor.org/info/rfc4724>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.

- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 5065, DOI 10.17487/RFC5065, August 2007, <<https://www.rfc-editor.org/info/rfc5065>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <<https://www.rfc-editor.org/info/rfc6514>>.
- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <<https://www.rfc-editor.org/info/rfc6793>>.

- [RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <<https://www.rfc-editor.org/info/rfc6811>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <<https://www.rfc-editor.org/info/rfc8177>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.

- [RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", RFC 8528, DOI 10.17487/RFC8528, March 2019, <<https://www.rfc-editor.org/info/rfc8528>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", RFC 8529, DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.
- [RFC9067] Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy", RFC 9067, DOI 10.17487/RFC9067, October 2021, <<https://www.rfc-editor.org/info/rfc9067>>.
- [RFC9127] Rahman, R., Ed., Zheng, L., Ed., Jethanandani, M., Ed., Pallagatti, S., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", RFC 9127, DOI 10.17487/RFC9127, October 2021, <<https://www.rfc-editor.org/info/rfc9127>>.
- [I-D.ietf-tcpm-yang-tcp] Scharf, M., Jethanandani, M., and V. Murgai, "A YANG Model for Transmission Control Protocol (TCP) Configuration", Work in Progress, Internet-Draft, draft-ietf-tcpm-yang-tcp-06, 3 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-tcpm-yang-tcp-06.txt>>.

10.2. Informative references

- [RFC3765] Huston, G., "NOPEER Community for Border Gateway Protocol (BGP) Route Scope Control", RFC 3765, DOI 10.17487/RFC3765, April 2004, <<https://www.rfc-editor.org/info/rfc3765>>.
- [RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <<https://www.rfc-editor.org/info/rfc5082>>.
- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", RFC 5925, DOI 10.17487/RFC5925, June 2010, <<https://www.rfc-editor.org/info/rfc5925>>.
- [RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", BCP 194, RFC 7454, DOI 10.17487/RFC7454, February 2015, <<https://www.rfc-editor.org/info/rfc7454>>.

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

Appendix A. Examples

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

A.1. Creating BGP Instance

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

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

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

```

    </route-flap-damping>
    <send-community>bt:standard</send-community>
    <send-community>bt:large</send-community>
    <description>"Peer Router B"</description>
    <afi-safis>
      <afi-safi>
        <name>bt:ipv4-unicast</name>
      </afi-safi>
    </afi-safis>
    <bfd>
      <enabled>true</enabled>
      <local-multiplier>2</local-multiplier>
      <desired-min-tx-interval>3300</desired-min-tx-interval>
    >
      <required-min-rx-interval>3300</required-min-rx-interv\
al>
    </bfd>
  </neighbor>
</neighbors>
</bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>

```

A.3. IPv6 Neighbor Configuration

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

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

```

<?xml version="1.0" encoding="UTF-8"?>
<key-chains
  xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
  <key-chain>
    <name>bgp-key-chain</name>
  </key-chain>
</key-chains>
<routing
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
  <control-plane-protocols>
    <control-plane-protocol>
      <type
        xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
type>
      <name>name:BGP</name>
      <bgp

```

```

        xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
    <global>
        <as>64496</as>
        <afi-safis>
            <afi-safi>
                <name
                    xmlns:bt=
                        "urn:ietf:params:xml:ns:yang:ietf-bgp-types">bt:ip\
v6-unicast</name>
                </afi-safi>
            </afi-safis>
        </global>
        <neighbors>
            <neighbor>
                <remote-address>2001:db8::</remote-address>
                <enabled>true</enabled>
                <secure-session-enable>true</secure-session-enable>
                <secure-session>
                    <enable-ao>true</enable-ao>
                    <ao-keychain>bgp-key-chain</ao-keychain>
                </secure-session>
                <peer-as>64497</peer-as>
                <description>"Peer Router B"</description>
                <timers>
                    <hold-time>120</hold-time>
                    <keepalive>70</keepalive>
                </timers>
            </neighbor>
        </neighbors>
    </bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>

```

A.4. VRF Configuration

This example shows how BGP can be configured for two VRFs, red and blue. In this case, the two network instances share a common AS, and distinguish between the instances using the router id.

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

```
<?xml version="1.0" encoding="UTF-8"?>
<network-instances
  xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
  <network-instance>
    <name>vrf-red</name>
    <vrf-root>
      <routing
        xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
        <router-id>192.0.2.1</router-id>
        <control-plane-protocols>
          <control-plane-protocol>
            <type
              xmlns:bgp=
                "urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</type\
>
              <name>BGP</name>
              <bgp
                xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
                <global>
                  <as>64496</as>
                  <afi-safis>
                    <afi-safi>
                      <name
                        xmlns:bt=
                          "urn:ietf:params:xml:ns:yang: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
                xmlns:bgp=
                  "urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</type\
```

```

>
    <name>BGP</name>
    <bgp
      xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
      <global>
        <as>64496</as>
        <afi-safis>
          <afi-safi>
            <name
              xmlns:bt=
                "urn:ietf:params:xml:ns:yang: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 version="1.0" encoding="UTF-8"?>
<routing-policy
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">
  <defined-sets>
    <prefix-sets>
      <prefix-set>
        <name>L0andL1</name>
        <mode>ipv4</mode>
        <prefixes>
          <prefix-list>
            <ip-prefix>1.1.1.1/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
          <prefix-list>
            <ip-prefix>2.2.2.2/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
        </prefixes>
      </prefix-set>
      <prefix-set>
        <name>L2andL3</name>
        <mode>ipv4</mode>
        <prefixes>
          <prefix-list>
            <ip-prefix>3.3.3.3/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
          <prefix-list>
            <ip-prefix>4.4.4.4/32</ip-prefix>
            <mask-length-lower>32</mask-length-lower>
            <mask-length-upper>32</mask-length-upper>
          </prefix-list>
        </prefixes>
      </prefix-set>
```

```
<prefix-set>
  <name>L4</name>
  <mode>ipv4</mode>
  <prefixes>
    <prefix-list>
      <ip-prefix>5.5.5.5/32</ip-prefix>
      <mask-length-lower>32</mask-length-lower>
      <mask-length-upper>32</mask-length-upper>
    </prefix-list>
  </prefixes>
</prefix-set>
</prefix-sets>
</defined-sets>
<policy-definitions>
  <policy-definition>
    <name>TO_R2</name>
    <statements>
      <statement>
        <name>10</name>
        <conditions>
          <match-prefix-set>
            <prefix-set>L0andL1</prefix-set>
          </match-prefix-set>
        </conditions>
        <actions>
          <bgp-actions
            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\
            <set-community>
              <options>add</options>
              <communities>11:11</communities>
            </set-community>
          </bgp-actions>
        </actions>
      </statement>
      <statement>
        <name>20</name>
        <conditions>
          <match-prefix-set>
            <prefix-set>L2andL3</prefix-set>
          </match-prefix-set>
        </conditions>
        <actions>
          <bgp-actions
            xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\
            <set-community>
              <options>remove</options>
```

```

        <communities>11:11</communities>
        <communities>22:22</communities>
      </set-community>
    </bgp-actions>
  </actions>
</statement>
<statement>
  <name>30</name>
  <conditions>
    <match-prefix-set>
      <prefix-set>L4</prefix-set>
    </match-prefix-set>
  </conditions>
  <actions>
    <bgp-actions
      xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp-policy">\
      <set-community>
        <options>replace</options>
        <communities>33:33</communities>
      </set-community>
    </bgp-actions>
  </actions>
</statement>
</statements>
</policy-definition>
</policy-definitions>
</routing-policy>

<routing
  xmlns="urn:ietf:params:xml:ns:yang:ietf-routing"
  xmlns:bt="urn:ietf:params:xml:ns:yang:ietf-bgp-types">
  <control-plane-protocols>
    <control-plane-protocol>
      <type
        xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp</\
type>
      <name>BGP</name>
      <bgp
        xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
        <global>
          <as>1</as>
          <afi-safis>
            <afi-safi>
              <name>bt:ipv4-unicast</name>
            </afi-safi>
          </afi-safis>
        </global>

```

```

    <neighbors>
      <neighbor>
        <remote-address>10.1.1.2</remote-address>
        <peer-as>2</peer-as>
        <afi-safis>
          <afi-safi>
            <name>bt:ipv4-unicast</name>
            </afi-safi>
          </afi-safis>
          <send-community>bt:standard</send-community>
          <apply-policy>
            <export-policy>TO_R2</export-policy>
            <default-export-policy>accept-route</default-export-po\
licy>
          </apply-policy>
        </neighbor>
      </neighbors>
    </bgp>
  </control-plane-protocol>
</control-plane-protocols>
</routing>

```

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 l3vpn-ipv4-unicast
|         +---rw prefix-limit
|         +---rw max-prefixes?                uint32
|         +---rw shutdown-threshold-pct?
|         |         rt-types:percentage
|         +---rw restart-timer?                uint32
+---rw l3vpn-ipv6-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
+---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 rcv-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 unreach-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)
      {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
  +--ro 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-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
+--ro active?                 boolean
+--ro prefixes
+--ro received?               uint32
+--ro sent?                   uint32
+--ro installed?              uint32
+--rw graceful-restart {bt:graceful-restart}?
+--rw enabled?                boolean
+--ro received?               boolean
+--ro advertised?             boolean
+--ro local-forwarding-state-preserved?
+--ro forwarding-state-preserved?
+--ro end-of-rib-received?    boolean
+--rw apply-policy
+--rw import-policy*          leafref
+--rw default-import-policy?  default-policy-type
+--rw export-policy*          leafref
+--rw default-export-policy?  default-policy-type
+--rw ipv4-unicast
+--rw prefix-limit
+--rw max-prefixes?           uint32
+--rw shutdown-threshold-pct?
+--rw restart-timer?          uint32
+--rw send-default-route?     boolean
+--rw ipv6-unicast
+--rw prefix-limit
+--rw max-prefixes?           uint32
+--rw shutdown-threshold-pct?
+--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 l3vpn-ipv4-unicast	+--rw prefix-limit	
	+--rw max-prefixes?	uint32
	+--rw shutdown-threshold-pct?	
	rt-types:percentage	
	+--rw restart-timer?	uint32
+--rw l3vpn-ipv6-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
+--rw use-multiple-paths	+--rw enabled?	boolean

```

|         +---rw ebgp
|         |         +---rw allow-multiple-as?    boolean
+---rw session-state?          enumeration
+---ro last-established?       yang:date-and-time
+---ro negotiated-capabilities* identityref
+---ro negotiated-hold-time?   uint16
+---ro last-error?            binary
+---ro fsm-established-time?   yang:gauge32
+---rw treat-as-withdraw?     boolean
+---ro erroneous-update-messages? uint32
+---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) {single-minimum-interval}?
|         |         +---rw min-interval?                uint32
+---rw statistics
|         +---ro peer-fsm-established-transitions?
|         |         yang:counter64
+---ro fsm-established-transitions?
|         yang:counter32
+---ro messages
|         +---ro in-total-messages?        yang:counter32
|         +---ro out-total-messages?       yang:counter32
|         +---ro in-update-elapsed-time?   yang:gauge32
+---ro sent
|         +---ro updates-received?        uint64
|         +---ro updates-sent?            uint64
|         +---ro messages-received?       uint64
|         +---ro messages-sent?           uint64
|         +---ro notification?            uint64
+---ro received
|         +---ro updates-received?        uint64
|         +---ro updates-sent?            uint64
|         +---ro messages-received?       uint64
|         +---ro messages-sent?           uint64
|         +---ro notification?            uint64
+---ro queues
|         +---ro input?                    uint32
|         +---ro output?                   uint32
+---x clear {bt:clear-statistics}?
|         +---w input
|         |         +---w clear-at?        yang:date-and-time
+---ro output
|         +---ro clear-finished-at?       yang:date-and-time

```



```

+---n established
|   +--- remote-address?   -> ../../neighbor/remote-address
|   +--- 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)?
|   |   |   +---:(operation-admin)
|   |   |   |   +---w admin?           empty
|   |   |   +---:(operation-hard)
|   |   |   |   +---w hard?            empty
|   |   |   +---:(operation-soft)
|   |   |   |   +---w soft?            empty
|   |   |   +---:(operation-soft-inbound)
|   |   |   |   +---w soft-inbound?    empty {bt:route-refresh}?
|   |   +---w clear-at?                yang:date-and-time
|   +---ro output
|   |   +---ro clear-finished-at?      yang:date-and-time
+---rw peer-groups
|   +---rw peer-group* [name]
|   |   +---rw name                    string
|   |   +---rw secure-session-enable?  boolean
|   |   +---rw secure-session
|   |   |   +---rw (option)?
|   |   |   |   +---:(ao)
|   |   |   |   |   +---rw enable-ao?          boolean
|   |   |   |   |   +---rw send-id?            uint8
|   |   |   |   |   +---rw rcv-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
|   |   |   |   +---:(ipsec)
|   |   |   |   |   +---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}?

```

```

+--rw enable?                boolean
+--rw suppress-above?        decimal64
+--rw reuse-above?           decimal64
+--rw max-flap?              decimal64
+--rw reach-decay?           uint32
+--rw unreach-decay?         uint32
+--rw keep-history?          uint32
+--rw send-community*        identityref
|                               {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)
|               {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
|   +--ro 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-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 l3vpn-ipv4-unicast
| | +--rw prefix-limit
| | | +--rw max-prefixes?          uint32
| | | +--rw shutdown-threshold-pct?
| | | |   rt-types:percentage
| | | +--rw restart-timer?        uint32
+--rw l3vpn-ipv6-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
+---rw interfaces
    |   +---rw interface* [name]
    |   |   +---rw name          if:interface-ref
    |   |   +---rw bfd {bt:bfd}?
    |   |   +---rw enabled?      boolean
+---ro rib
    |   +---ro attr-sets
    |   |   +---ro attr-set* [index]
    |   |   |   +---ro index          uint64
    |   |   |   +---ro attributes
    |   |   |   |   +---ro origin?
    |   |   |   |   |   bt:bgp-origin-attr-type
    |   |   |   |   +---ro atomic-aggregate?      boolean
    |   |   |   |   +---ro next-hop?              inet:ip-address
    |   |   |   |   +---ro link-local-next-hop?    inet:ipv6-address
    |   |   |   |   +---ro med?                    uint32
    |   |   |   |   +---ro local-pref?              uint32
    |   |   |   |   +---ro originator-id?          yang:dotted-quad
    |   |   |   |   +---ro cluster-list*           yang:dotted-quad
    |   |   |   |   +---ro aigp-metric?            uint64
    |   |   |   +---ro aggregator
    |   |   |   |   +---ro as?          inet:as-number
    |   |   |   |   +---ro address?     inet:ipv4-address
    |   |   |   +---ro aggregator4
    |   |   |   |   +---ro as4?        inet:as-number
    |   |   |   |   +---ro address?     inet:ipv4-address
    |   |   |   +---ro as-path
    |   |   |   |   +---ro segment* []
    |   |   |   |   |   +---ro type?      identityref
    |   |   |   |   |   +---ro member*    inet:as-number
    |   |   |   +---ro as4-path
    |   |   |   |   +---ro segment* []
    |   |   |   |   |   +---ro type?      identityref
    |   |   |   |   |   +---ro member*    inet:as-number
    |   |   +---ro communities
    |   |   |   +---ro community* [index]
    |   |   |   |   +---ro index          uint64
    |   |   |   |   +---ro community*    union
    |   |   +---ro ext-communities
    |   |   |   +---ro ext-community* [index]
    |   |   |   |   +---ro index          uint64
    |   |   |   |   +---ro ext-community*  rt:route-target
    |   +---ro large-communities

```

```

|   +--ro large-community* [index]
|   |   +--ro index          uint64
|   |   +--ro large-community*  bt:bgp-large-community-type
+--ro afi-safis
|   +--ro afi-safi* [name]
|   |   +--ro name          identityref
|   |   +--ro ipv4-unicast
|   |   |   +--ro loc-rib
|   |   |   |   +--ro routes
|   |   |   |   |   +--ro route* [prefix origin path-id]
|   |   |   |   |   |   +--ro prefix
|   |   |   |   |   |   |   inet:ipv4-prefix
|   |   |   |   |   |   +--ro origin          union
|   |   |   |   |   |   +--ro path-id          uint32
|   |   |   |   |   |   +--ro attr-index?      leafref
|   |   |   |   |   |   +--ro community-index?  leafref
|   |   |   |   |   |   +--ro ext-community-index? leafref
|   |   |   |   |   |   +--ro 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
|   |   |   |   |   |   |   |   |   boolean
|   |   |   |   |   |   |   |   +--ro ineligible-reason?

```



```

+--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

```

```

|         |         +---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 optional?          boolean
    +--ro transitive?       boolean
    +--ro partial?          boolean
    +--ro extended?         boolean
    +--ro attr-type         uint8
    +--ro attr-len?         uint16
    +--ro attr-value?       binary
  +--ro reject-reason?      union
+--ro clear-routes {bt:clear-routes}?
  +---x clear
    +---w input
      +---w clear-at?
        yang:date-and-time
  +--ro output
    +--ro clear-finished-at?
      yang:date-and-time
+--ro adj-rib-out-post
+--ro routes
  +--ro route* [prefix path-id]
    +--ro prefix
      |
      inet:ipv6-prefix
    +--ro path-id
      uint32
    +--ro attr-index?       leafref
    +--ro community-index?  leafref
    +--ro ext-community-index? leafref
    +--ro 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

```



```

    |   |   +---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

```

```
    |   +--rw (method)?
    |   |   +--:(inline)
    |   |   |   +--rw communities*                rt-types:route-target
    |   |   +--:(reference)
    |   |   |   +--rw ext-community-set-ref?    leafref
    +--rw set-large-community
    |   +--rw options?
    |   |   bgp-set-community-option-type
    +--rw (method)?
    |   +--:(inline)
    |   |   +--rw communities*
    |   |   |   bt:bgp-large-community-type
    +--:(reference)
    |   +--rw large-community-set-ref?    leafref
```

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

IDR
Internet-Draft
Intended status: Standards Track
Expires: July 14, 2022

F. Qin
China Mobile
H. Yuan
UnionPay
T. Zhou
G. Fioccola
Y. Wang
Huawei
January 10, 2022

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."

This Internet-Draft will expire on July 14, 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	2
2. Motivation	3
3. IFIT methods for SR Policy	4
4. IFIT Attributes in SR Policy	5
5. IFIT Attributes Sub-TLV	6
5.1. IOAM Pre-allocated Trace Option Sub-TLV	8
5.2. IOAM Incremental Trace Option Sub-TLV	9
5.3. IOAM Directly Export Option Sub-TLV	9
5.4. IOAM Edge-to-Edge Option Sub-TLV	10
5.5. Enhanced Alternate Marking (EAM) sub-TLV	11
6. SR Policy Operations with IFIT Attributes	12
7. IANA Considerations	12
8. Security Considerations	13
9. Acknowledgements	14
10. References	14
10.1. Normative References	14
10.2. Informative References	16
Appendix A.	16
Authors' Addresses	16

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:

IOAM: the reference documents for the data plane are
[I-D.ietf-ippm-ioam-ipv6-options] for SRv6 and
[I-D.gandhi-mpls-ioam-sr] for SR-MPLS.

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:

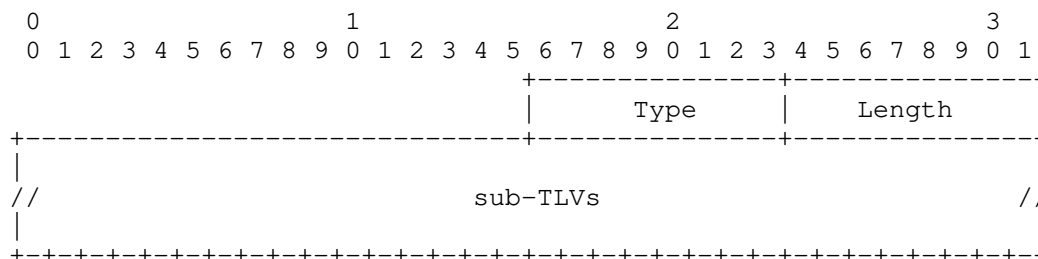


Fig. 1 IFIT Attributes Sub-TLV

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:

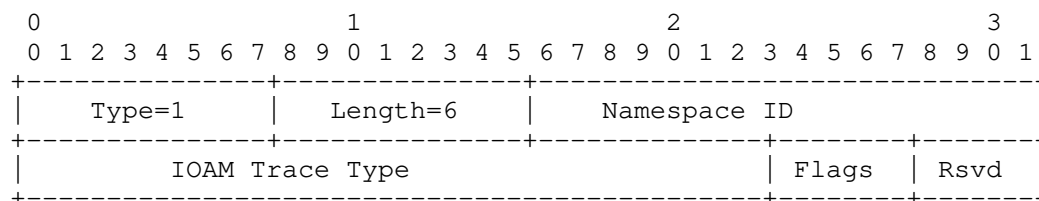


Fig. 2 IOAM Pre-allocated Trace Option Sub-TLV

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:

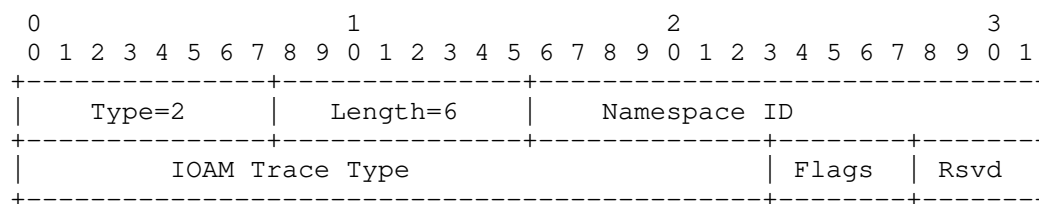


Fig. 3 IOAM Incremental Trace Option Sub-TLV

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:

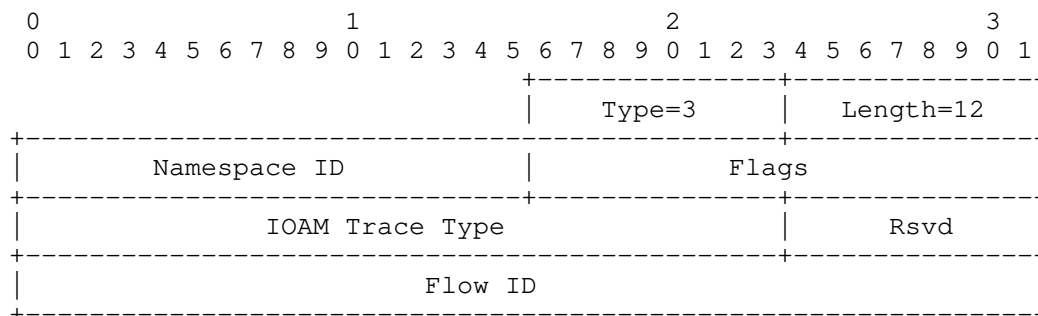


Fig. 4 IOAM Directly Export Option Sub-TLV

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:

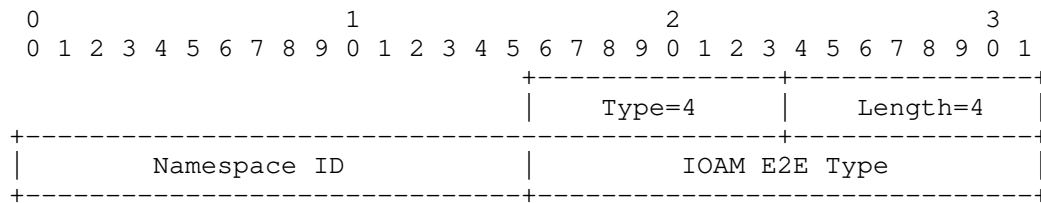


Fig. 5 IOAM Edge-to-Edge Option Sub-TLV

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-Namespaces. 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:

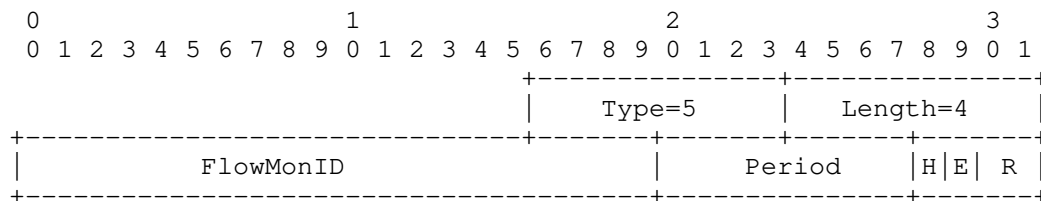


Fig. 6 Enhanced Alternate Marking Sub-TLV

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:

Codepoint	Description	Reference
TBD1	IFIT Attributes Sub-TLV	This document

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:

Value	Description	Reference
1	IOAM Pre-allocated Trace Option Sub-TLV	This document
2	IOAM Incremental Trace Option Sub-TLV	This document
3	IOAM Directly Export Option Sub-TLV	This document
4	IOAM Edge-to-Edge Option Sub-TLV	This document
5	Enhanced Alternate Marking Sub-TLV	This document

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 controlled 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., Velde, 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.

- [I-D.ietf-ippm-ioam-flags]
Mizrahi, T., Brockners, F., Bhandari, S., Sivakolundu, R., Pignataro, C., Kfir, A., Gafni, B., Spiegel, M., and J. Lemon, "In-situ OAM Loopback and Active Flags", draft-ietf-ippm-ioam-flags-07 (work in progress), October 2021.
- [I-D.ietf-ippm-ioam-ipv6-options]
Bhandari, S. and F. Brockners, "In-situ OAM IPv6 Options", draft-ietf-ippm-ioam-ipv6-options-06 (work in progress), July 2021.
- [I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-14 (work in progress), October 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

[RFC8799] Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", RFC 8799, DOI 10.17487/RFC8799, July 2020, <<https://www.rfc-editor.org/info/rfc8799>>.

10.2. Informative References

- [I-D.chen-pce-pcep-ifit]
Yuan, H., Zhou, T., Li, W., Fioccola, G., and Y. Wang, "Path Computation Element Communication Protocol (PCEP) Extensions to Enable IFIT", draft-chen-pce-pcep-ifit-04 (work in progress), July 2021.
- [I-D.gandhi-mpls-ioam-sr]
Gandhi, R., Ali, Z., Filsfils, C., Brockners, F., Wen, B., and V. Kozak, "MPLS Data Plane Encapsulation for In-situ OAM Data", draft-gandhi-mpls-ioam-sr-06 (work in progress), February 2021.
- [I-D.gandhi-mpls-rfc6374-sr]
Gandhi, R., Filsfils, C., Voyer, D., Salsano, S., and M. Chen, "Performance Measurement Using RFC 6374 for Segment Routing Networks with MPLS Data Plane", draft-gandhi-mpls-rfc6374-sr-05 (work in progress), June 2020.
- [I-D.ietf-mpls-rfc6374-sfl]
Bryant, S., Swallow, G., Chen, M., Fioccola, G., and G. Mirsky, "RFC6374 Synonymous Flow Labels", draft-ietf-mpls-rfc6374-sfl-10 (work in progress), March 2021.

Appendix A.

Authors' Addresses

Fengwei Qin
China Mobile
No. 32 Xuanwumenxi Ave., Xicheng District
Beijing
China

Email: qinfengwei@chinamobile.com

Hang Yuan
UnionPay
1899 Gu-Tang Rd., Pudong
Shanghai
China

Email: yuanhang@unionpay.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Giuseppe Fioccola
Huawei
Riesstrasse, 25
Munich
Germany

Email: giuseppe.fioccola@huawei.com

Yali Wang
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: wangyalil1@huawei.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 1 July 2022

K. Vairavakkalai
M. Jeyanthan
Juniper Networks, Inc.
28 December 2021

BGP signalled MPLS-namespaces
draft-kaliraj-bess-bgp-sig-private-mpls-labels-04

Abstract

The MPLS-forwarding-layer in a core network is a shared resource. The MPLS FIB at nodes in this layer contains labels that are dynamically allocated and locally significant at that node.

For some usecases like upstream-label-allocation, it is useful to be able to create virtual private MPLS-forwarding-layers over this shared MPLS-forwarding-layer. This allows installing deterministic private label-values in the private-FIBs created at nodes participating in this private MPLS forwarding-layer, while preserving the "locally significant" nature of the underlying shared 'public' MPLS-forwarding-layer.

This specification describes the procedures to create such virtual private MPLS-forwarding layers (private MPLS-planes) using a new BGP family. And gives a few example use-cases on how this private forwarding-layers can be used.

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 1 July 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
2. Motivation	3
3. Constructs and building blocks	4
3.1. Context Protocol Nexthop Address	4
3.2. MPLS context FIB	4
3.3. Context Label	5
3.4. Roles of nodes in a MPLS-plane	5
3.4.1. Edge-nodes (PLER)	5
3.4.2. Transit-nodes (PLSR)	5
3.5. Sending traffic into the MPLS plane	6
4. Terminology	6
5. BGP families, routes and encoding	7
5.1. New address-families for "MPLS namespace signaling" . . .	8
5.1.1. AFI: MPLS, SAFI: 128	8
5.1.2. AFI: MPLS, SAFI: 1	8
5.2. Routes and Operational procedures	9
5.2.1. "Context-Nexthop" discovery route	9
5.2.2. MPLS namespace "Private Label" routes	10
6. Example of Usecases	13
6.1. Mezanine transport layer in a Seamless-MPLS network . . .	13
6.2. Service Forwarding Helper usecase	14
6.3. Standard BGP API to a MPLS network's forwarding-plane . .	14
6.4. Traffic engineering and Security advantages	14
7. IANA Considerations	15
8. Security Considerations	15
9. Acknowledgements	15
10. Normative References	15
Authors' Addresses	16

1. Introduction

The MPLS-forwarding-layer in a core network is a shared resource. The MPLS FIB at nodes in this layer contains labels that are dynamically allocated and locally significant at that node.

For some usecases like upstream-label-allocation, it is useful to be able to create virtual private MPLS-forwarding-layers over this shared MPLS-forwarding-layer. This allows installing deterministic private label-values in the private-FIBs in this private forwarding-layer, while preserving the "locally significant" nature of the underlying shared 'public' MPLS-forwarding-layer.

It can be noted that, mechanism described in this document is nothing but a [RFC4364] style BGP VPN where the FEC is MPLS-Label, instead of IP-prefix. This document defines new address-families (AFI: MPLS, SAFI: VPN-Unicast, Unicast) and associated signaling mechanisms to create and use MPLS forwarding-contexts in a network. The concepts of MPLS-Context-tables and upstream allocation are described in [RFC5331].

BGP speakers participating in the private MPLS FIB layer create instances of "MPLS forwarding-context" FIBs, which are identified using a "Context-Protocol-Nexthop (CPNH)". A Context-label MAY be advertised in conjunction with the Context Protocol Nexthop (CPNH) using new BGP address-family to other speakers.

2. Motivation

A provider's core network consists of a global-domain (default forwarding-tables in P and PE nodes) that is shared by all tenants in the network and may also contain multiple private user-domains (e.g. VRF route tables).

The global MPLS forwarding-layer can be viewed as the collection of all default MPLS forwarding-tables. This global MPLS Fib layer contains labels locally significant to each node. The "local-significance of labels" gives the nodes freedom to participate in MPLS-forwarding with whatever label-ranges they can support in forwarding hardware.

In emerging usecases some applications using the MPLS-network may benefit from a "static labels" view of the MPLS-network. In some other usecases, a standard mechanism to do Upstream label-allocation is beneficial.

It is desirable to leave the global MPLS FIB layer intact, and build private MPLS FIB-layers on top of it to achieve these requirements. The private-MPLS-FIBs can then be used by the applications as desired. The private MPLS-FIBs need to be created only at the nodes in the network where predictable label-values (external label allocation) is desired. E.g. P-routers that need to act as a "Detour-nodes" or "Service-Forwarding-Helpers" that need to mirror service-labels.

In other words, provisioning of these private MPLS-FIBs can be gradual and can co-exist with nodes not supporting the feature described in this document. These private-MPLS-FIBs can be stitched together using either the Context-labels over the existing shared MPLS-network tunnels, or 'private' context-interfaces - to form the "private MPLS-FIB layer".

An application can then install the routes with desired label-values in the private forwarding-contexts with desired forwarding-semantics.

3. Constructs and building blocks

The building-blocks that construct a private MPLS plane are described in this section.

3.1. Context Protocol Nexthop Address

A private MPLS plane (just "MPLS plane" here-after) is identified by an IP-address called Context Protocol Nexthop (CPNH). This address is unique in the core-network, like any other loopback address.

A loopback-address uniquely identifies a specific node in the network, and we call it Global Protocol Nexthop (GPNH) in this document. The CPNH address uniquely identifies a "MPLS-plane".

Each node that has forwarding-context for a MPLS-plane MUST be configured with the same CPNH but a different RD, such that the RD:CPNH will uniquely identify that node in the MPLS-plane.

3.2. MPLS context FIB

An instance of a MPLS forwarding-table at a node in the private MPLS-plane. This Private MPLS FIB contains the private-label routes.

A node can have context-FIB for multiple MPLS-planes. The same label-value can have a different forwarding-semantic in each MPLS-plane. Thus the applications using that MPLS-plane get a deterministic label-value independent of other applications using other MPLS-planes.

The terms "private MPLS FIB-layer" and "private MPLS-plane" are used interchangeably in this document.

3.3. Context Label

A context-label is a non-reserved dynamically allocated label, that is installed in the global MPLS FIB, and points to a MPLS-Context-FIB. The Context-Label have forwarding semantics as follows in the global MPLS-FIB:

Context-Label -> Pop and Lookup in MPLS-Context-Fib

Advertising the "Context-Label in conjunction with the GPNH" tells the network how to reach a "RD:CPNH".

3.4. Roles of nodes in a MPLS-plane

The node roles in a MPLS-plane can be classified into "edge nodes" (call them PLER) or "transit-nodes" (call them PLSR).

3.4.1. Edge-nodes (PLER)

Private Label Edge-routers (PLER) have MPLS context-FIB that belong to the MPLS-plane. They advertise the presence of this context-FIB using transport layer address families like BGP-CT [BGP-CT] or BGP-LU, and private-label routes from this FIB are advertised using new BGP AFI/SAFI described in this document.

3.4.2. Transit-nodes (PLSR)

These are just Border-nodes that do label-swap forwarding for the Context-Labels they see in the Context-Protocol-Nexthop advertisement routes (BGP-CT or BGP-LU) going thru them. They basically stitch/extend the label switched path to a PLER's CPNH when they re-advertise the CPNH routes with nexthop-self.

PLSRs dont have MPLS context-FIBs. PLSRs dont have Context Protocol-Nexthop. Because they dont have Private label routes to originate.

However a node in the network can play both roles, of PLER and PLSR.

3.5. Sending traffic into the MPLS plane

At a PLER, MPLS-traffic arriving with private-label hits the correct private MPLS-FIB by virtue of either arriving on a "private network-interface" that is attached to the MPLS context-FIB, or arriving with a "Context-label" on a network-interface attached to the global MPLS-FIB.

To send data traffic into this private MPLS plane, the sender MUST use as handle either a "Context-label" advertised by a node or a "Private-interface" owned by the MPLS context-FIB at the node. The MPLS context-FIB is created for an application that needs a private MPLS-plane.

The Context-Label is the only dynamic label-value the application needs to learn from the network (PLER node it is connected to), to be able to use the private MPLS-plane. The application can chose predictable value for the labels to be programmed in the private MPLS-FIBs.

Once the packet enters the private MPLS plane at an edge-node (PLER), the node will forward the packet to the next node (PLSR or PLER), by pushing the Context-label advertised by that next-node, and the transport-label to reach that node's GPNH. This will repeat until the packet reaches the PLER's private MPLS-FIB that originated that private MPLS-label.

At each PLER in the MPLS-plane, the private-label value remains the same, and points towards the same resource attached to the MPLS-plane. This allows the applications using the MPLS-network a static-labels view of the resources attached to the private MPLS-plane.

At each PLSR in the MPLS-plane, the context-label value will change (be swapped in forwarding), but is transparent to the application.

4. Terminology

P-router : A Provider core router, also called a LSR

LSR : Label Switch Router (pure transport node speaking LDP, RSVP etc)

PLSR: a BGP-CT or BGP-LU transit node in a private MPLS-plane, that does label-swap forwarding for Context-Label.

PLER: an edge node in a private MPLS-plane. It has a forwarding-context for private-labels.

Detour-router : A BGP border node that is used as a loose-hop in a traffic-engineered path

PE-router : Provider Edge router, that hosts a service (Internet, L3VPN etc)

SE-router : Service Edge router. Same as PE.

SFH-router : Service Forwarding Helper. A node helping an SE-router with service-traffic forwarding, using Service-routes mirrored by the SE.

MPLS FIB : MPLS Forwarding table

Global MPLS FIB : Global MPLS Forwarding table, to which shared-interfaces are connected

Private MPLS FIB : Private MPLS Forwarding table, to which private-interfaces are connected

Private MPLS FIB Layer (Private MPLS plane): The group of Private MPLS FIBs in the network, connected together via Context-Labels

Context-Label : Locally-significant Non-reserved label pointing to a private MPLS FIB

Context nexthop IP-address (CPNH) : An IP-address that identifies the "Private MPLS FIB Layer". RD:CPNH identifies a Private MPLS FIB at a specific BGP node.

Global nexthop IP-address (GPNH) : Global Protocol Nexthop address. E.g. a loopback address used as transport tunnel end-point.

5. BGP families, routes and encoding

This section describes the new constructs defined by this document.

5.1. New address-families for "MPLS namespace signaling"

This document defines a new AFI: "MPLS" (IANA code TBD). And two new address-families, using SAFIs 128 and 1. These address families are used to signal "MPLS namespaces" in BGP. To send or receive routes of these address families, these AFI, SAFI pair of values MUST be negotiated in Multiprotocol Extensions capability described in RFC4760 [RFC4760]

5.1.1. AFI: MPLS, SAFI: 128

This address-family is used to exchange private label-routes in private MPLS-FIBs at routers that are connected using a common network interface. The private label route has NLRI prefix format "RD:PrivateLabel" and contains Route-Target extended-community identifying the private-FIB-Layer (VPN) the route belongs to. The nexthop of these routes is set to either the GPNH or the CPNH of the BGP-speaker advertising the RFC-8277 label.

Any transport layer protocol is used to advertise the Context-Label that the receiving router uses to send traffic into the private MPLS-FIB. The Context-Label installed in the global MPLS-FIB points to the private MPLS-FIB. The Context-Label is required when the connecting-interface is a shared common interface that terminates into the global MPLS FIB.

Routes of this address-family can be sent with either IPv4 or IPv6 nexthop. The type of nexthop is inferred from the length of the 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).

When the length of Next Hop Address field is 12 the nexthop address is of type VPN-IPv4 with 8-octet RD.

5.1.2. AFI: MPLS, SAFI: 1

This address-family is used to exchange private label-routes in private MPLS-FIBs to routers that are connected using a private network-interface.

Because the interface is private, and terminates directly into the private MPLS-FIB, a Context-Label is not required to access the private MPLS-FIB and NLRI prefix format is just "PrivateLabel/24", without the RD.

Routes of this address-family can be sent with either IPv4 or IPv6 nexthop. The type of nexthop is inferred from the length of the nexthop.

When the length of Next Hop Address field is 16 (or 32) the nexthop address is of type IPv6 (potentially followed by the link-local IPv6 address of the next hop).

When the length of Next Hop Address field is 4 the nexthop address is a 4 octet IPv4 address.

5.2. Routes and Operational procedures

5.2.1. "Context-Nexthop" discovery route

The Context-NH discovery route may be a BGP-LU or [BGP-CT] family route that carries CPNH in the "Prefix" portion of the NLRI. And the Context-Label is carried in the "Label" field in the [RFC8277] format NLRI.

This route is advertised with the following path-attributes:

- * BGP Nexthop attribute (code 14, MP_REACH) carrying GPNH address.
- * Route-Target extended community, identifying the Transport class, if applicable.

The "Context-Nexthop discovery route" is originated by each speaker who acts as a PLER. The "RD:Context-nexthop" uniquely identifies the private-MPLS-FIB at the speaker. The "Context-nexthop address" uniquely identifies the private-MPLS-plane in the network. The Context-Label advertised in this route has a local forwarding semantic of "Pop, Lookup in Private-MPLS-FIB".

A BGP speaker readvertising a BGP-CT Context-Nexthop for RD:CPNH discovery-route MUST follow the mechanisms described in [BGP-CT]. Specifically when re-advertising with "next-hop self" MUST allocate a new Label with a forwarding semantic of "Swap Received-Context-Label, Forward to Received-GPNH". This extends reachability to the CPNH across tunnel domains.

5.2.2. MPLS namespace "Private Label" routes

The Private Label routes are carried in the new address-family "MPLS VpnUnicast" (AFI:MPLS, SAFI:128) aka "MPLS-namespace signaling", defined in this document.

The NLRI format follows the specifications in [RFC8277], with the "Prefix" portion of the NLRI comprising of the RD and "Private MPLS Label" encoded as shown below.

In a MP_REACH_NLRI attribute whose AFI/SAFI is MPLS/128, the "Length" field will be 112 bits or less, comprising of the Label, RD and "Private MPLS Label".

In a MP_REACH_NLRI attribute whose AFI/SAFI is MPLS/1, the "Length" field will be 48 bits or less, comprising of the Label, and "Private MPLS Label".

NLRI Prefix (Private Label route, AFI:MPLS, SAFI:128)

This picture shows NLRI format when the RFC-8277 Multiple Labels Capability is not used:

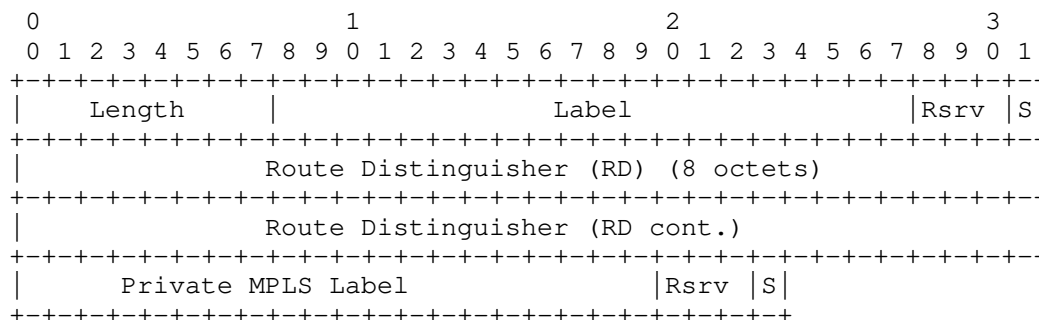


Fig 1: RFC-8277 NLRI with one Label.

- Length:

The Length field consists of a single octet. It specifies the length in bits of the remainder of the NLRI field.

In a MP_REACH_NLRI attribute whose AFI/SAFI is MPLS/128, the "Length" field will be 112 bits or less, comprising of the Label, RD and "Private MPLS Label".

As specified in [RFC4760], the actual length of the NLRI field will be the number of bits specified in the Length field,

rounded up to the nearest integral number of octets.

- Label:
The Label field is a 20-bit field containing an MPLS label value (see [RFC3032]). This label is locally significant, downstream allocated at the speaker identified in the BGP Nexthop field in MP_REACH_NLRI (code 14). This label is pushed in nexthop of the route installed in MPLS context FIB at receiving router.
- Route Distinguisher (RD):
The 8 byte Route Distinguisher as specified in [RFC4760].
- Private MPLS Label:
The "Private MPLS Label" field is a 20-bit field containing an MPLS label value (see [RFC3032]). This is an upstream assigned MPLS label, used as destination of route installed in MPLS context FIB at the receiving router.
- Rsrv:
This 3-bit field SHOULD be set to zero on transmission and MUST be ignored on reception.
- S:
This 1-bit field MUST be set to one on transmission and MUST be ignored on reception.

Attributes on this route:

- * BGP Nexthop attribute (code 14, MP_REACH) carrying a GPNH address.
(OR)
- * The Multi-nexthop attribute [MULTI-NH] with forwarding-semantic:
 - "Forward to RD:CPNH"
- * Route-Target extended-community, identifying the private FIB-layer

MultiNexthop BGP-attribute (Private Label route)

MultiNH.Num-Nexthops = 1
FwdSemanticsTLV.FwdAction = Forward
NHDescrTLV.NhopDescrType = RD:CPNH or GPNH

Fig 2: MultiNexthop attr of Private Label route

A speaker MAY readvertise a private-label-route without changing the Nexthop (RD:CPNH) carried in it, if the speaker is a pure PLSR.

If it does alter the nexthop to SelfRD:CPNH, it SHOULD act as a PLER, and for e.g. originate a "Context-Nexthop discovery route" for prefix "SelfRD:CPNH".

Even if the speaker sets nexthop-address to Self because of regular BGP readvertisement-rules, Label Prefix MUST NOT be altered, and the received NLRI "RD:Private-Label1" MUST be re-advertised as-is. Such that value of label "Private-Label1" doesn't change while the packet traverses multiple nodes in the private-MPLS-FIB-layer.

The Route-target attached to the route is the one identifying the private MPLS FIB layer (VPN). The Private-label routes resolve over the Context-nexthop route that belong to the same VPN.

A node receiving a "Private-Label route" RD:L1 MUST install the label L1 in the private MPLS Forwarding-context identified by the Route-Target attached to the route.

The label route MUST be installed with forwarding-semantic as specified in the received Multi-nexthop attribute. As an example, a Detour node MAY receive the private-label-route with a forwarding-semantic of "Forward to RD:CPNH" operation. And an Egress node MAY receive a private-label-route with a forwarding-semantic pointing to a resource it houses. Note that such a Private-label BGP-route MAY be received from external-application also.

5.2.2.1. Resolving received Private Label-routes

A node receiving a "Context-nexthop discovery route" MUST be capable of using either the CPNH or the RD:CPNH carried in the NLRI, to resolve other routes received with this CPNH address or RD:CPNH in the "Nexthop-attributes".

The receiver of a private-label route MUST recursively resolve the received nexthop (RD:CPNH) over the Context-Nexthop discovery-route for prefix "RD:CPNH" to determine the label stack "Context-Label, Transport-Label" to push, so that the MPLS packet with private-label reaches the private MPLS FIB originating the route.

If a node receives multiple "Context-nexthop discovery route" for a CPNH, it SHOULD run path-selection after stripping the RD, to find the closest ingress to the private-MPLS-plane identified by the CPNH. This best path SHOULD be used to resolve a received private-label-route.

6. Example of Usecases

6.1. Mezanine transport layer in a Seamless-MPLS network

Typically service-routes in a MPLS network bind to the following entities that identify point-of-presence of a service:

- * Protocol Nexthop - PE loopback address (GPNH)
- * Service Label - PE advertised locally significant label that identifies the service

In this model, whenever a PE is taken out of service the GPNH changes, and Service-Label changes - which causes maintenance a heavy convergence event. Because the service-routes with massive-scale need to be readvertised with new service-label or PE-address.

An alternate model could be: to advertise the Service-routes with a protocol-nexthop of CPNH (without RD), with a forwarding-semantic of:

- * "Push <Private-Label>, and Forward to CPNH"

This model fully decouples the service-layer from the transport-layer identifiers, by making the Service-routes refer to the CPNH and Private-Labels. Thus the underlying transport-layer can change (nodes representing a Private-label can be added or removed) without any changes to the service-routes. Which present good scaling properties for the network.

This model also allows anycast traffic forwarding to any resource in the network. Multiple PEs can advertise the same Private-Label to identify a specific service (e.g. peering with an AS) they are offering.

Once the service-route traffic enters the private-FIB-layer, at the closest entry-point determined by path-selection of CPNH auto-discovery routes; then the Private-Labels (with pre-determined values) pushed will determine the loose hop path taken by the traffic and also the destination-resource.

6.2. Service Forwarding Helper usecase

In a virtualized environment a Service-PE node (that comprises of a vCP and multiple vFPs) can mirror MPLS labels (GL1) in its global MPLS-FIB to a private forwarding context at an upstream node (SFH) with information on which vFPs are optimal exit-points for that label. Such that the SFH can optimally forward traffic to GL1 to the right vFPs, thus avoiding intra fabric traffic hops.

To do this, the service-PE advertises a private-label route with RD:GL1 to the SFH node. The route is advertised with a Multi-nexthop attribute with one or more legs that have a "Forward to SEPx" semantics. Where SEPx is one of many exit-points at the Service-PE node.

6.3. Standard BGP API to a MPLS network's forwarding-plane

This mechanism facilitates predictable (external-allocator determined) label-values, using a standard BGP-family as the API. It gives the external applications a separate MPLS-FIB to play with, totally separate from other applications.

This also avoids vendor specific-API dependencies for external-allocators (controller softwares), and vice-versa.

This mechanism also increases the overall MPLS label-space available in the network, because it creates per-app label-forwarding-contexts (namespaces), instead of reserving/splitting the global MPLS FIB among various applications.

6.4. Traffic engineering and Security advantages

- * Ability of ingress to steer mpls-traffic thru specific detour loose-hop nodes using predictable-labels' stack.
- * Provide label-spoofing protection at edge-nodes - by virtue of using separate mpls-forwarding-contexts
- * Allow private-MPLS label usage to spread across multiple-domains/ AS and work seamlessly with existing technologies like Inter-AS VPN option C.

7. IANA Considerations

This document makes following requests of IANA.

New BGP AFI code ("Address Family Numbers" registry):

* 16399 for "MPLS Namespaces"

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Using separate mpls-forwarding-contexts for separate applications and stitching them into separate MPLS-planes increases the security attributes of the MPLS network.

9. Acknowledgements

The authors thank Jeffrey (Zhaohui) Zhang, Ron Bonica, Jeff Haas and John Scudder for the valuable discussions.

10. Normative References

- [BGP-CT] Vairavakkalai, K., "BGP Classful Transport Planes", 25 August 2021, <<https://tools.ietf.org/html/draft-kaliraj-idr-bgp-classful-transport-planes-12#section-11.3>>.
- [MULTI-NH] Vairavakkalai, K., "BGP MultiNexthop attribute", 28 December 2021, <<https://tools.ietf.org/html/draft-kaliraj-idr-multinexthop-attribute-04>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.

[RFC5331] Aggarwal, R., Rekhter, Y., and E. Rosen, "MPLS Upstream Label Assignment and Context-Specific Label Space", RFC 5331, DOI 10.17487/RFC5331, August 2008, <<https://www.rfc-editor.org/info/rfc5331>>.

[RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.

Authors' Addresses

Kaliraj Vairavakkalai
Juniper Networks, Inc.
1133 Innovation Way,
Sunnyvale, CA 94089
United States of America

Email: kaliraj@juniper.net

Minto Jeyananth
Juniper Networks, Inc.
1133 Innovation Way,
Sunnyvale, CA 94089
United States of America

Email: minto@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 October 2022

K. Vairavakkalai, Ed.
N. Venkataraman
B. Rajagopalan
Juniper Networks, Inc.
G. Mishra
Verizon Communications Inc.
M. Khaddam
Cox Communications Inc.
X. Xu
Capitalonline.
R. Szarecki
Google.
J. Gowda
Extreme Networks
Yadlapalli
ATT
26 April 2022

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 administered 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.

Table of Contents

1. Introduction	3
2. Terminology	5
3. Transport Class	6
4. "Transport Class" Route Target Extended Community	7
5. Transport RIB	9
6. Transport Routing Instance	9
7. Nexthop Resolution Scheme	9
8. BGP Classful Transport Family NLRI	10
9. Comparison with other families using RFC-8277 encoding	11
10. Protocol Procedures	12
11. Scaling considerations	15
11.1. Avoiding unintended spread of CT routes across domains.	15
11.2. Constrained distribution of PNHS to SNs (On Demand Nexthop)	16
11.3. Limiting scope of visibility of PE loopback as PNHS	17
12. OAM considerations	17
13. Applicability to Network Slicing	18
14. SRv6 support	19
15. Illustration of procedures with example topology	19
15.1. Topology	19
15.2. Service Layer route exchange	21
15.3. Transport Layer route propagation	21
15.4. Data plane view	23
15.4.1. Steady state	23
15.4.2. Absorbing failure of primary path	24
16. IANA Considerations	25
16.1. New BGP SAFI	25
16.2. New Format for BGP Extended Community	25
16.2.1. Existing registries to be modified	25
16.2.2. New registries to be created	26
16.3. MPLS OAM code points	27
17. Security Considerations	27
18. Contributors	27
19. Acknowledgements	28
20. Normative References	28
Authors' Addresses	30

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 nexthop 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 readvertisements 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.

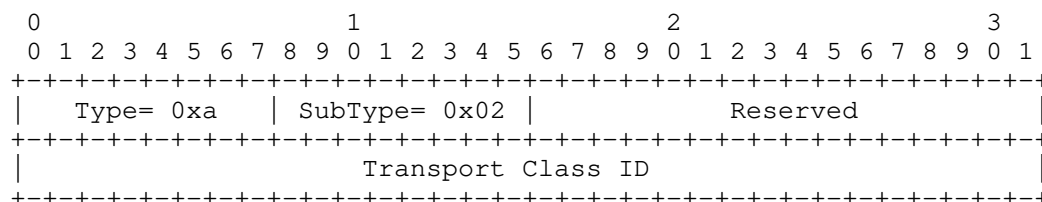
Similarly, PCEP-RSVP-COLOR [PCEP-RSVP-COLOR] extends PCEP to carry RSVP Color. This color association thus learnt is also mapped to a Transport Class thus associating the PCEP signaled RSVP LSP with the desired Transport Class.

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:



"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 Constrain 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 readvertising 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] is 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 propagation 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:

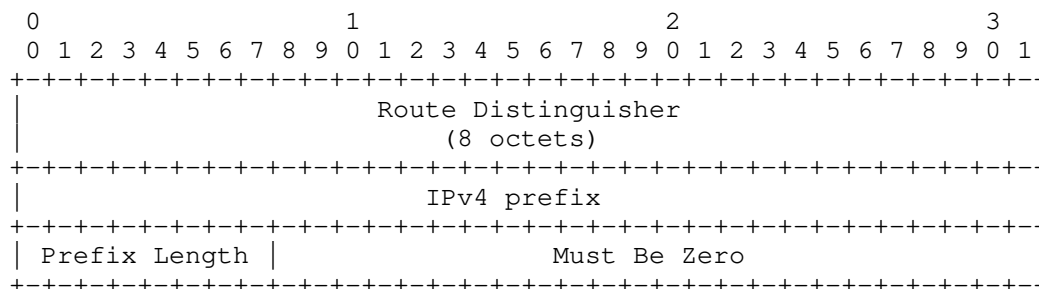


Figure 1: Classful Transport IPv4 FEC

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:

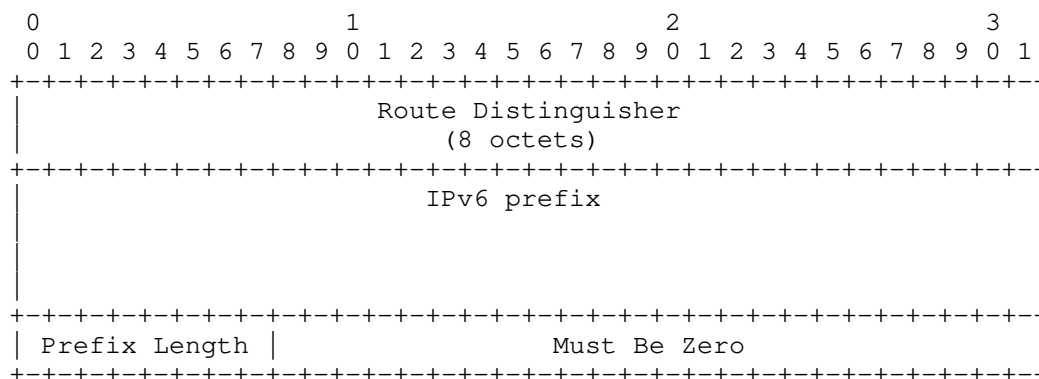


Figure 2: Classful Transport IPv6 FEC

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.

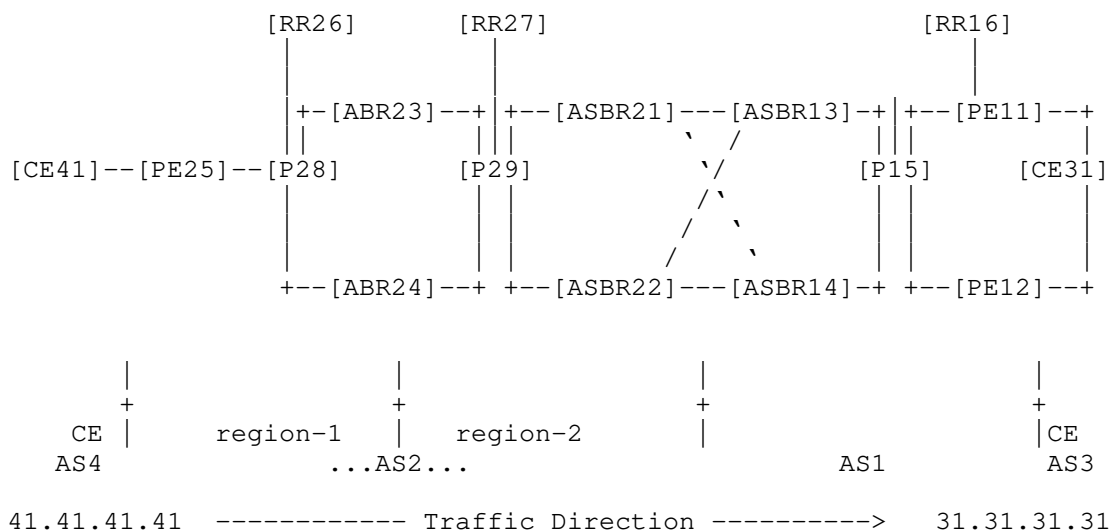
RFC8986, [SRV6-INTER-DOMAIN] specify the SRv6 Endpoint behaviors (End USD, End.BM, End.B6.Encaps and End.Replace, End.ReplaceB6, respectively). These are leveraged for BGP CT with SRv6 data plane.

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



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_ASBR22_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_ASBR21_bronze - RSVP tunnel
- ABR23_to_ASBR22_bronze - RSVP tunnel
- ABR24_to_ASBR21_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 readvertises 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 readvertises 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 readvertises 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 readvertises 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 tunneled 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_ASBR22_gold tunnel. ABR23 readvertises 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_ASBR22_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 yields 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

	TYPE VALUE	NAME
+	0x0a	Transitive Transport Class Extended
+		Community (Sub-Types are defined in the
+		"Transitive Transport Class Extended
+		Community Sub-Types" registry)

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

	TYPE VALUE	NAME
+	0x4a	Non-Transitive Transport Class Extended
+		Community (Sub-Types are defined in the
+		"Non-Transitive Transport Class Extended
+		Community Sub-Types" registry)

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

RANGE	REGISTRATION PROCEDURE
0x00-0xBF	First Come First Served
0xC0-0xFF	IETF Review

SUB-TYPE VALUE	NAME
0x02	Route Target

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

RANGE	REGISTRATION PROCEDURE
0x00-0xBF	First Come First Served
0xC0-0xFF	IETF Review
SUB-TYPE VALUE	NAME
0x02	Route Target

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, Shradha Hegde, Richard Roberts, Krzysztof Szarkowicz, John E Drake, Srihari 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]

Vairavakkalai, Ed., "BGP signalled MPLS-namespaces", 11 June 2021, <<https://tools.ietf.org/html/draft-kaliraj-bess-bgp-sig-private-mpls-labels-01#section-6.1>>.

[PCEP-RSVP-COLOR]

Rajagopalan, Ed., "Path Computation Element Protocol (PCEP) Extension for RSVP Color", 15 January 2021, <<https://datatracker.ietf.org/doc/html/draft-rajagopalan-pcep-rsvp-color-00>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

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

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.

[RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", RFC 4456, DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.

- [RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <<https://www.rfc-editor.org/info/rfc4684>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC7153] Rosen, E. and Y. Rekhter, "IANA Registries for BGP Extended Communities", RFC 7153, DOI 10.17487/RFC7153, March 2014, <<https://www.rfc-editor.org/info/rfc7153>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.
- [RFC8029] Kompella, K., Swallow, G., Pignataro, C., Ed., Kumar, N., Aldrin, S., and M. Chen, "Detecting Multiprotocol Label Switched (MPLS) Data-Plane Failures", RFC 8029, DOI 10.17487/RFC8029, March 2017, <<https://www.rfc-editor.org/info/rfc8029>>.
- [RFC8212] Mauch, J., Snijders, J., and G. Hankins, "Default External BGP (EBGP) Route Propagation Behavior without Policies", RFC 8212, DOI 10.17487/RFC8212, July 2017, <<https://www.rfc-editor.org/info/rfc8212>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <<https://www.rfc-editor.org/info/rfc8664>>.
- [RFC8669] Previdi, S., Filsfils, C., Lindem, A., Ed., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix Segment Identifier Extensions for BGP", RFC 8669, DOI 10.17487/RFC8669, December 2019, <<https://www.rfc-editor.org/info/rfc8669>>.

- [RTC-Ext] Zhang, Z., Ed., "Route Target Constrain Extension", 12 July 2020, <<https://tools.ietf.org/html/draft-zzhang-idr-bgp-rt-constrains-extension-00#section-2>>.
- [Seamless-SR] Hegde, Ed., "Seamless Segment Routing", 17 November 2020, <<https://datatracker.ietf.org/doc/html/draft-hegde-spring-mpls-seamless-sr-03>>.
- [SRTE] Previdi, S., Ed., "Advertising Segment Routing Policies in BGP", 18 November 2019, <<https://tools.ietf.org/html/draft-ietf-idr-segment-routing-te-policy-08>>.
- [SRV6-INTER-DOMAIN] K A, Ed., "SRv6 inter-domain mapping SIDs", 10 January 2021, <<https://datatracker.ietf.org/doc/html/draft-salih-spring-srv6-inter-domain-sids-00>>.
- [SRV6-MPLS-AGRWL] Agrawal, Ed., "SRv6 and MPLS interworking", 22 February 2021, <<https://datatracker.ietf.org/doc/draft-agrawal-spring-srv6-mpls-interworking/05/>>.
- [SRV6-SERVICES] Dawra, Ed., "SRv6 BGP based Overlay Services", 11 April 2021, <<https://datatracker.ietf.org/doc/html/draft-ietf-bess-srv6-services-07>>.

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

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 25 July 2022

J.M. Jeganathan
V.S.K.R. Avula
Juniper Networks
21 January 2022

BGP Peer Auto-Configuration
draft-minto-idr-bgp-autodiscovery-01

Abstract

This document describes a layer 3 protocol (Service advertisement) to help bgp to advertise service availability and local configurations . This enables bgp speakers to discover bgp peers transport endpoints and peer's configuration within link. With Service advertisement, receivers could successfully bring up bgp protocol session without mundane configurations.

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 25 July 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
1.1. Requirements Language	3
2. Protocol overview	3
3. PUD layers	3
4. Messages	5
4.1. SA Base message	5
4.1.1. Remaining lifetime TLV	5
4.1.2. Config sequence TLV	5
4.1.3. Authentication TLV	6
4.1.4. Refresh request TLV	6
4.2. BGP service advertisement message	6
4.2.1. Local address	7
4.2.2. Local IPv6 address	8
4.2.3. Security TTL	8
4.2.4. Security Authentication	8
4.2.5. TCP MSS	8
4.2.6. Link Address	9
5. Protocol operation	9
5.1. Transmit procedure	10
5.2. Receiver procedure	10
5.3. Transport endpoint reachability	11
5.4. Protocol Authentication operation	11
6. Acknowledgements	12
7. IANA Considerations	12
7.1. Message of SA	12
7.2. TLVs of SA base Message	13
7.3. TLVs of BGP service advertisement message	13
8. Security Considerations	13
9. References	14
9.1. Normative References	14
9.2. Informative References	14
Appendix A. Additional Stuff	14
Authors' Addresses	14

1. Introduction

This document describes a layer 3 protocol (Service advertisement) to help bgp to advertise service availability and local configurations. This enables bgp speakers to discover bgp peer's transport endpoints and peer's configuration within link. With Service advertisement, receivers could successfully bring up bgp protocol session without mundane configurations.

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. Protocol overview

This is a simple protocol to periodically send and receive UDP multicast PDU that contains bgp transport information in the form of messages and TLVs. Receiver could use this information to bootstrap the single hop bgp and/or loopback address bgp between directly connected bgp speakers. The advertised information gets expired if it is not refreshed before the lifetime ends.

This protocol does not provide any reliability of delivery and relies on UDP multicast and periodic send. The current version of this protocol assumes the link MTU is good enough to encode BGP transport information or underlying IP implementation is able to fragment and reassemble for link local multicast PDU. But this protocol is flexible enough to implement a future version of fragment TLV attachment. This is to bypass smaller link MTU for a system or environment preventing IP fragment.

Service Advertisement (SA) PDU has multiple types of messages. This document defines 2 types of messages. The primary/base messages are required for SA to operate and secondary type messages for BGP service advertisement.

3. PUD layers

The PDU contains a header followed by variable number of messages. Each message contains variable number of TLVs.

SA uses type-length-value format.

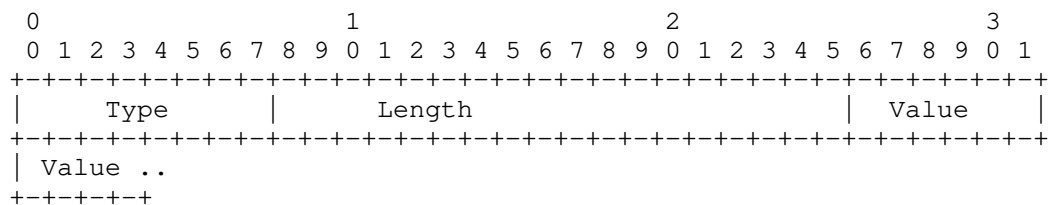


Figure 1

Type: 1-octet value to interpret the value with in message. Same type value could be reused in different message.

Length: Specifies length in octets of the value field.

Value: Octet string that encodes information to be interpreted as specified by the Type field.

SA uses message to group set of TLVs

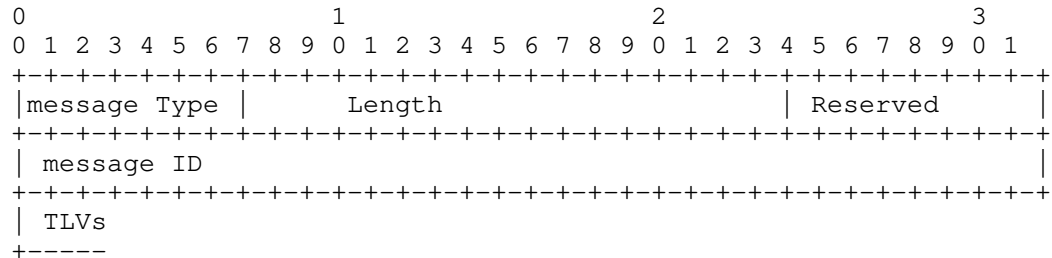


Figure 2

message Type: This 1-octet value identifies type of message.

Message Length: Specifies the length in octets of the Message ID and TLVs.

Message ID :32-bit value used to identify this message. Used for logging purpose.

SA PDU

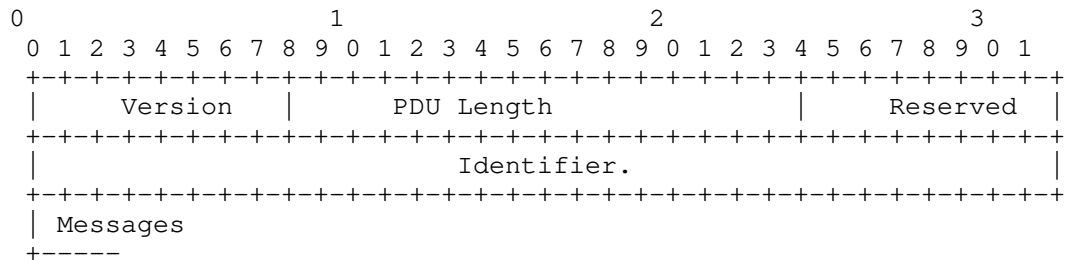


Figure 3

Version: This 1-octet unsigned integer indicates the protocol version. This version of the specification specifies Service Advertisement version 0.

PDU Length: This 2-octet unsigned integer specifies the length

Identifier: 4 octet field that uniquely identifies PDU sender. BGP id could be used for this purpose. This helps to uniquely identify sender across the parallel links between same nodes.

4. Messages

The document defines following messages.

1. SA Base message
2. BGP service advertisement message

4.1. SA Base message

The SA Base message is mandatory message and mainly used for the protocol operation.

The document defines following TLVs for SA Base message.

1. Remaining lifetime TLV
2. Config sequence TLV
3. Authentication TLV
4. Refresh request TLV

4.1.1. Remaining lifetime TLV

Remaining lifetime describes how long receiver should keep the state without seeing a PDU from the sender. The lifetime gets updated when receiver accepts the PDU.

Type : 17

Length: 2 octets

Value: Remaining lifetime in seconds

4.1.2. Config sequence TLV

Specifies a 4-octet configuration sequence number. Receiver could make use the number to detect config change. This will be useful to restart the bgp session with new parameters.

Type : 18

Length: 4 octets

Value: unsigned sequence number

4.1.3. Authentication TLV

Specifies authentication.

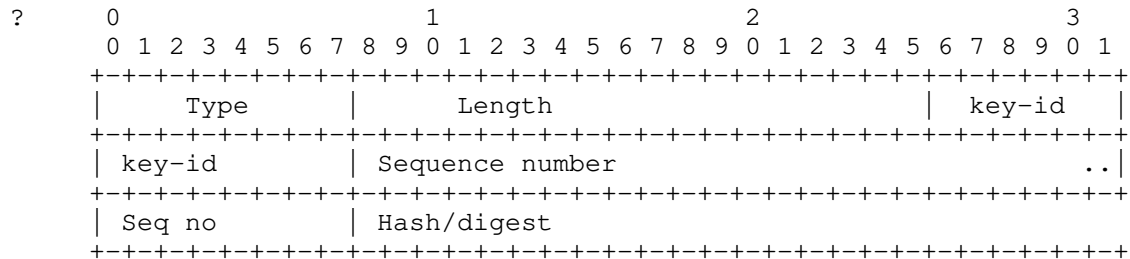


Figure 4

Type : 19

Length: variable

key-id : keychain id

Sequence-number - 4 byte sequence number of this SA base message.

Digest - Hash computed for this message using key-id mapped algorithm

4.1.4. Refresh request TLV

Optional TLV to trigger receivers to immediately send SA PDU. Presence of the TLV indicates sender request refresh. This will be used during the restart to learn about services quickly from connected devices to speed up service discovery.

Type : 20

Length: 0 octets

4.2. BGP service advertisement message

BGP Service Advertisement message provides transport information to bring up the bgp session. This document defines transport information TLVs and session information TLVs for BGP Service Advertisement messages.

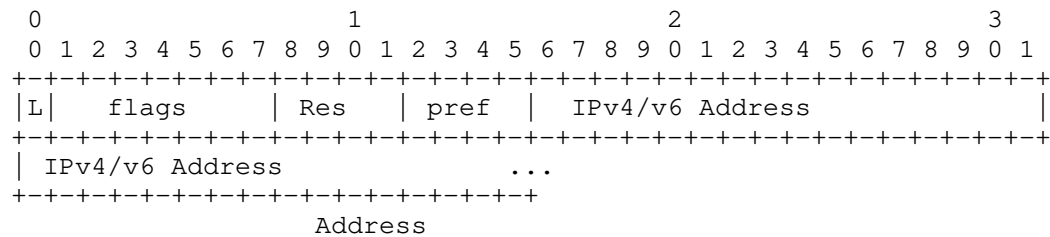
Message type of BGP Service Advertisement message: 2.

Following are the Transport information TLVs

1. Local Address
2. Security TTL
3. Security Authentication
4. Link Address
5. Transport Preference.
6. TCP MSS

4.2.1. Local address

Specifies a local address used for bgp transport connection. Address encoding uses a below format. 2 octets describe the address and followed by address value.



L bit - Address of loopback interface.

pref - Preference value of 4 bits. Value from 1 to 15.

0 indicates dont care.

1 highly preferred and 15 means least preferred.

Address - For IPv4 4 octets and IPv6 16 octets

Figure 5

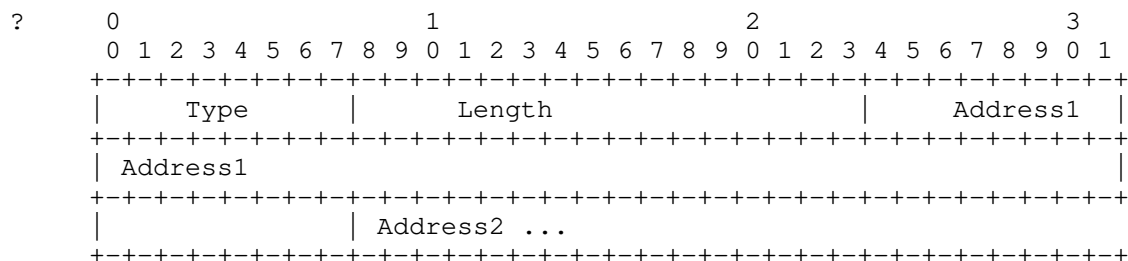


Figure 6

Type : 17

Length: variable. multiples of 6 octets

Value: IPv4 addresses with address encoding.

4.2.2. Local IPv6 address

Specifies a IPv6 local address used for bgp transport connection.

Type : 18

Length: multiples of 18 octets

Value: IPv6 addresses with address encoding used for transport connection.

4.2.3. Security TTL

TTL be accepted for bgp messages

Type : 19

Length: 0

Value: Presence of this TLV indicates that receiver accepts only packets with 255 TTL.

4.2.4. Security Authentication

Type : 20

Length: 1

Value: This supports only two values 0 and 1.

0 indicates TCP md5. 1 indicates TCP-AO Absence of this TLV indicates, no authentication used for connection.

4.2.5. TCP MSS

TCP MSS used for the connection

Type : 21

Length: 4

Value: Value in bytes

Indicates the preference of TCP MSS for the transport connection.

4.2.6. Link Address

This could be used for receiver to get nexthop information for local address TLV when sender's running IPv4 PDU and prefer IPv6 transport and vice-a-versa. This could also be used to provide reachability to loopback addresses with link address.

Type : 22

Length: 4 for IPv4 address and 16 for IPv6 address 6 for mac address.

Value: interface IPv4 or IPv6 or mac address .

5. Protocol operation

A sender should periodically send PDU to refresh the advertised information before its lifetime expires. An implementation may send PDU well before the lifetime expires based on specific events. These events could be a local config change or discovering a new advertiser. Also, implementation could switch to fast refresh when content of the pdu changes and move back to regular refresh interval. The fast refresh will help in quicker discovery and may help update content in case of auto order delivery. As stated above, this is purely an implementation technique than the protocol mandate.

To discover multi-data(IPv4/IPv6) protocol environment (mixed transport mode in a single link) sender shall send both data-protocol pdu based on local configuration. When sender choose to send both data protocol PDU it should make sure that semantic content of the messages should be same. An implementation may choose to use preferred data protocol PDU as primary send PDU and only send other data protocol PDU during the interesting events. This optimization is only possible when all the known advertisers participates in both data-protocol.

A sender should send PDU to refresh before previously advertised lifetime expires. If bgp is configured with only one transport address family(IPv4/v6) then sender shall only send corresponding data protocol PDU. If both addresses are configured, then it shall use both data protocol PDUs. PDUs are sent with source address as link primary address and destination is link local all- routers with TTL 255. If authentication is enabled then add authentication TLV using the authentication procedure described in authentication section. Populate other TLVs based on local preference and send the PDU on configured link. Semantic content (transport and session information) of the PDU should be same irrespective of data protocol.

Receiver resets the state when it accepts new PDU irrespective of the data protocol. Receiver shall add a route for the address in local address TLV with nexthop as source address of the PDU if PDU(PUD) data protocol and local address is same address family. Otherwise if link address is available, it could be used as nexthop for the address in local address TLV. Receivers consolidate state from various TLVs and pass it on to BGP for the session opening. An implementation could only notify if the state change from previous reported state to bgp or the configuration sequence number changes from the receiver. How bgp uses this information is beyond the scope of the document.

5.1. Transmit procedure

PDUs are sent with source address as link's primary address and destination is link local all-routers with TTL 255. PDU is sent to SA UDP port(179 if assigned). After the header, SA Base message should be first message. If authentication is enabled then add authentication TLV using the authentication procedure described in authentication section. This authentication TLV should be first TLV of PDU. Add lifetime and config sequence TLVs defined in this document. Both these TLVs are mandatory TLVs. After the SA base message, add bgp service advertisement message with appropriate TLVs.

5.2. Receiver procedure

When a SA PDU received, following sanity procedure must be followed.

If TTL is not 255 then discard the PDU.

If the version is not compatible (Only compatible version is 0) then discard the PDU.

If the PDU length is greater than IP header length, then discard it.

If the first message is not SA Base, then discard the pdu.

If authentication is enabled and first TLV in the SA base message, then discard the PDU.

If authentication is enabled, then follow the authentication procedure.

If authentication is failed, then discard the PDU.

With above steps, sanity of the PDU header is verified. Receiver should start decoding the TLV information. Once all the TLV sanity checked receiver shall keep the decoded information. If the receiver decides to keep the information, then it should start a timer with specified lifetime or refresh lifetime with newer one.

The identifier in PDU header uniquely identifies the advertisement. An implementation could either implement neighbor semantic or state semantic from the advertised information along with identifier. This document does not recommend one or other.

Received and decoded information shall be passed on to bgp if the content does not match with last received or the local config has changed. This is a desired optimization, so that SA does not unnecessarily trigger failed bgp session open attempts. How bgp uses this information is beyond the scope of the document.

5.3. Transport endpoint reachability

Advertised local address reachability can either be gathered from the source address or a link address TLV. Source address of the PDU may not give reachability for all deployment (Sender using the IPv6 data protocol but prefer v4 transport). In those cases, link address TLV will provide reachability.

5.4. Protocol Authentication operation

A sender that wants to authenticate Service messages should include Authentication TLV as part of SA base message.

Sender needs to include all the fields of Authentication TLV as shown in section 4.1.3. It needs to assign a unique KEY-ID to each authentication combination configured on the device. Key-length needs to be set to configured key's length in bytes. Sequence number is a 32-bit unsigned integer that may increment by one each time a new message is sent. Any change in TLVs for a previously advertised local address needs to be sent with an incremented TLV. Digest value can be of variable length depending upon type of authentication being used. This value is calculated over all the contents of service message.

Receiver on receiving this TLV has a sequential processing of individual fields of TLV. Sequence number is read from TLV and is compared against any existing state from this sender. If sequence number is lesser than previously received, this packet is dropped except when bgp session goes down. If last received sequence number was m and current received sequence number is n , n needs to be in range of $[m+1, m + 2^{(32 - 1)}]$. This exception is needed to handle a

restarted sender who is unable to retrieve earlier sequence number due to restart. This is required when SA uses bigger lifetime. After getting KEY-id, it checks for a matching KEY-ID on it. If it does not exist, packet is dropped. Next Key-length of locally configured key is compared against key-length received in this TLV, if they do not match packet is dropped. Similarly, a comparison is done for authentication types of locally configured key and received TLV. If they do not match, packet is dropped. After above checks, hash is computed for all the contents of service with locally configured key and compared against received hash value. If they are same, authentication information matches with local configuration and messages can be further processed with protocol operations depending on type of this message.

6. Acknowledgements

Jeffrey Hass provided many useful technical and editorial comments and suggestions for improvement.

7. IANA Considerations

This document requests IANA to allocate a new UDP port (179 is the preferred number) and 2 message type code for service advertisements.

Value TLV Name Reference -----

```

----- Service Name:
Service advertisements Transport Protocol: UDP Assignee: IESG
iesg@ietf.org Description: Service advertisements for auto
configuration. Reference: This document --
draft-minto-idr-bgp-autodiscovery.txt Port Number: 179 -- To be
assigned by IANA.
```

Figure 7

7.1. Message of SA

This document requests IANA to create a new registry following messages "Messages of SA " with the following registration procedure:

Registry Name: Messages of SA protocol			
?	Value	Message name	Reference
	-----	-----	-----
	0	Reserved	This document
	1	Base message	This document
	2	BGP Service Advertisement	This document

Figure 8

7.2. TLVs of SA base Message

This document requests IANA to create a new registry following messages "TLVs of SA base Message" with the following registration procedure:

Registry Name: TLVs of SA base Message.		Reference
Value	TLV Name	
0-16	Reserved	This document
17	Remaining lifetime TLV	This document
18	Config sequence TLV	This document
19	Authentication	This document
20	Refresh request TLV	This document
224-255	Experimental	

Figure 9

7.3. TLVs of BGP service advertisement message

This document requests IANA to create a new registry following messages "TLVs of BGP Service Advertisement" with the following registration procedure:

?	Registry Name: TLVs of BGP Services.		Reference
	Value	TLV Name	
	0-16	Reserved	This document
	17	Local Address	This document
	18	Local IPv6 Address	This document
	19	Security TTL	This document
	20	Security Authentication	This document
	21	TCP MSS	This document
	22	Link Address	This document
	224-255	Experimental	This document

Figure 10

8. Security Considerations

This security considerations for BGP [RFC4271] apply equally to this extension for BGP session establishment.

BGP sessions transport end points discovered over this protocol can be protected against various attacks by using authentication for packets as described in Section 5.4.

Usage of sequence number and authentication reduces likelihood of replay attacks. As the protocol is not connection-oriented, it makes it feasible to change authentication parameters for protocol messages. This further reduces the likelihood of replay-attacks.

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

9.2. Informative References

- [bgp-autoconf-considerations] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://datatracker.ietf.org/doc/draft-ietf-idr-bgp-autoconf-considerations/>>.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <<https://www.rfc-editor.org/info/rfc2629>>.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <<https://www.rfc-editor.org/info/rfc3552>>.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, 2006, <<https://www.rfc-editor.org/rfc/rfc4271>>.
- [RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <<https://www.rfc-editor.org/info/rfc5226>>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Jeyananth Minto Jeganathan
Juniper Networks
Juniper Networks, 1133 Innovation Way
Sunnyvale, CA 94089
United States of America

Email: minto@juniper.net

Venkata Shiva Krishna Reddy Avula
Juniper Networks
Juniper Networks, 1133 Innovation Way
Sunnyvale, CA 94089
United States of America

Email: venkatashiva@juniper.net

IDR
Internet-Draft
Intended status: Standards Track
Expires: July 27, 2022

S. Sangli
S. Hegde
R. Das
Juniper Networks Inc.
B. Decraene
Orange
January 23, 2022

Generic Metric for the AIGP attribute
draft-ssangli-idr-bgp-generic-metric-aigp-02

Abstract

This document defines extensions to the AIGP attribute to carry Generic Metric sub-types. This is applicable when multiple domains exchange BGP routing information. The extension will aid in intent-based end-to-end path selection.

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 27, 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	2
2. Requirements Language	3
3. Multiple Metric types	4
4. Issues with RFC7311	4
5. Generic Metric TLV	5
6. Usage of Generic-Metric TLV	6
7. Updates to Decision Procedure	7
8. Use-case: Different Metrics across Domains	8
9. Deployment Considerations	9
10. Contiguity Compliance	10
11. Backward Compatibility	10
12. Security Considerations	10
13. IANA Considerations	11
14. Acknowledgements	11
15. References	11
15.1. Normative References	11
15.2. Informative References	12
Authors' Addresses	13

1. Introduction

Large Networks belonging to an enterprise may consist of nodes in the order of thousands and may span across multiple IGP domains where each domain can run separate IGPs or levels/areas. BGP may be used to interconnect such IGP domains, with one or more IGP domains within an Autonomous System. The enterprise network can have multiple Autonomous Systems and BGP may be employed to provide connectivity between these domains. Furthermore, BGP can be used to provide routing over a large number of such independent administrative domains.

The traffic types have evolved over years and operators have resorted to defining different metric types within a IGP domain (ISIS or OSPF) for IGP path computation. An operator may want to create an end-to-end path that satisfy certain intent. The intent could be to create end-to-end path that minimizes one of the metric-types. Some metrics can be assigned administratively by an operator and they are described in the base ISIS, OSPF specifications. Other metrics, for example, are the Traffic Engineering Default Metric defined in [RFC5305] and [RFC3630], Min Unidirectional delay metric defined in [RFC8570] and [RFC7471]. There may be other metrics such as jitter, reliability, fiscal cost, etc. that an operator may wish to express

as the cost of a link. The procedures mentioned in the above specifications describe the IGP path computation within IGP domains.

With the advent of 5G applications and Network Slicing applications, an operator may wish to provision end-to-end paths across multiple domains to cater to traffic constraints. This is also known as intent-based inter-domain routing and there are certain architectures being developed as described in [I-D.hegde-spring-seamless-sr-architecture] and [I-D.dskc-bess-bgp-car-problem-statement] . The Classful Transport Planes as described in [I-D.kaliraj-idr-bgp-classful-transport-planes] and Color-Based Routing as described in [I-D.dskc-bess-bgp-car] describe how end-to-end intent-based paths can be established. The proposal described in this document can be used in conjunction with such architectures.

When multiple domains are interconnected via BGP, protocol extensions for advertising best-external path and/or ADDPATH as described in [RFC7911] are employed to take advantage of network connectivity thus providing alternate paths. The Color-Based Routing and Classful Transport Planes routing proposals describe approaches that result in alternate paths for a reaching one destination. During the BGP best path computation, the step(e) as per section 9.1.2.2 of [RFC4271] , the interior cost of a route as determined via the IGP metric value can be used to break the tie. In a network spanning multiple IGP domains, the AIGP TLV encoded within the AIGP attribute described in [RFC7311] can be used to compute the AIGP-enhanced interior cost to be used in the decision process for selecting the best path as documented in section 2 of [RFC7311] . The [RFC7311] specifies how AIGP TLV can carry the accumulated IGP metric value.

There is a need to synchronize the metric-type values carried between IGP and BGP in order to avoid operational overhead of translation between them. The existing AIGP TLV carries a TLV type and metric-value where TLV type does not map to IGP metric-types defined in the IGP metric-type registry. Hence there is a need to provide a generic metric template to embed the IGP metric-type values within the AIGP attribute. This document extends the AIGP attribute for carrying Generic-Metric TLV and the well-defined sub metric types. This document also provides procedures for handling Generic-Metric during the BGP best path computation.

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. Multiple Metric types

Consider the network as shown in Figure 1. The network has multiple domains. Each domain runs a separate IGP instance. Within each domain iBGP sessions are established between the PE routers. eBGP sessions are established between the Border Routers across domains. An operator wishes to compute end-to-end path optimized for a metric-type delay. Each domain will be enabled to compute the IGP paths based on metric-type delay. Such values should also be propagated to the adjacent domains for effective end-to-end path computation.

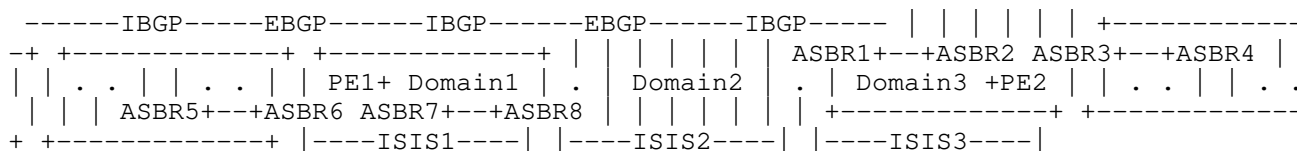


Figure 1: WAN Network

The AIGP TLV in the AIGP attribute as specified in [RFC7311] supports the IGP default metric. If all domains use IGP cost as the metric, then one can compute the end-to-end path with shortest IGP cost. However if an operator wishes to compute the end-to-end path with metric other than IGP cost, we need additional extensions to the AIGP attribute for carry the metric-types and metric values.

The [I-D.ietf-lsr-flex-algo-bw-con] proposes a generic metric type that can embed multiple metric types within it. It supports both standard metric-types and user-defined metric-types. This document leverages the generic-metric draft and proposes extensions to the AIGP attribute to carry Generic Metric TLV as specified below.

4. Issues with RFC7311

The following procedures are not clearly described in [RFC7311] .

- o The section 3 describes "When an AIGP attribute is created, it SHOULD contain no more than one AIGP TLV. However, if it contains more than one AIGP TLV, only the first one is used as described in Sections 3.4 and 4. In the remainder of this document, we will use the term value of the AIGP TLV to mean the value of the first AIGP TLV in the AIGP attribute. Any other AIGP TLVs in the AIGP attribute MUST be passed along unchanged if the AIGP attribute is passed along."
- oOne MUST interpret that more than one TLV of a particular type (i.e. AIGP TLV metric-type 1) can be present in the update and only the first occurrence MUST be analysed. All other TLVs (type 2

or type 3 etc.) MUST be passed along unchanged if AIGP attribute is passed along.

- o The section 3.2 describes "Note that an AIGP attribute MUST NOT be considered to be malformed because it contains more than one TLV of a given type or because it contains TLVs of unknown types."
- oOne MUST interpret that opaque TLVs (TLVs with type 2 or type 3 for example) MUST be passed along if ADVERTISE_AIGP_ATTRIBUTE has been enabled to a neighbor.
- o Section 3.3 describes "The AIGP attribute MUST NOT be sent on any BGP session for which AIGP_SESSION is disabled."
- oWhile maintaining the non-transitivity is important, it is also important to provide accumulated cost end-to-end across domains. If there are more than one TLVs in the AIGP attribute, it becomes important to define the behavior of which TLV gets updated and sent across domains.
- o The rules for route redistribution is not clearly described.
- oWhen a BGP route is redistributed, should AIGP metric-value be used directly as the cost in IGP or should there be a policy to modify AIGP metric-value before redistributing the route into IGP. It is important to define the behavior of route redistribution metric conversion when redistribution occurs on multiple domains along the path.

5. Generic Metric TLV

This document proposes a new TLV : Generic-Metric TLV in the AIGP attribute. This will carry the metric type and metric value used in the network. The format is shown below.

```

0 1 2 3 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 2 +-+--+
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
type | +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
value | +--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

Figure 2: Generic-Metric TLV

Generic-Metric TLV Type (1 octet): Code point to be assigned by IANA

Generic-Metric TLV Length (2 octets): 12

Generic-Metric TLV Value (9 octets): 2 sub-fields as shown below:

1. metric-type (1 octet): Value from IGP metric-type registry.

2. metric-value (8 octets): Value range (0 - 0xffffffffffffffff)

6. Usage of Generic-Metric TLV

1. When a BGP speaker wishes to generate AIGP attribute with Generic-Metric TLV for a prefix, it MUST perform the following procedures.

- * The procedures specified in [RFC7311] section 3.4 should be followed that describes creation of attribute, modifications by the originator and non-originator of the route.
- * Repeated metric changes may cause large number of BGP updates to get generated and be propagated throughout the network. In order to avoid that, a configurable threshold is defined. If the difference between the new metric-value and the advertised metric-value is less than the configured threshold, the update MAY be suppressed. If the new metric-value is above the configured threshold, a new BGP update containing the new metric-value SHOULD be advertised.
- * If the domain uses a metric type other than IGP cost for the IGP path computation, the BGP speaker MAY add Generic-Metric TLV to the AIGP attribute before advertising to a neighboring BGP speaker.
- * The metric-type sub-field in the Generic-Metric TLV will carry the value indicating the type of the metric as specified in the IGP metric-type registry.
- * The value of the metric or cost to reach the prefix being advertised will be encoded in the metric-value sub-field. This is the cost or the distance to the destination prefix from the advertising BGP speaker which sets itself as the next hop as described in section 3.4 of [RFC7311] .
- * Procedures for defining the cost to reach a next hop for various metric-types is outside the scope of this document.

2. When a BGP speaker wishes to send a BGP update attaching the AIGP attribute, it must validate if that session has been enabled for sending the AIGP attribute as per procedures mentioned in [RFC7311] .

3. When a BGP speaker receives a BGP update that has a route to T with next hop N and has the AIGP attribute with Generic-Metric TLV it MUST perform the following procedures.

- * It must validate if that session has been enabled to receive the AIGP attribute as per rules mentioned in [RFC7311] .
- * If the BGP speaker does not recognize the Generic-Metric TLV or type of metric encoded in metric-type subfield of the TLV, then the BGP speaker will ignore the Generic-Metric TLV and follow the BGP decision procedure as specified in [RFC7311] .
- * If the metric-type of the path used for resolving the next hop N matches with the metric-type of Generic-Metric TLV of the AIGP attribute, then the metric-value sub-field MUST be used in the AIGP-enhanced interior cost computation as specified in the next section.
- * If the metric-type of the path used for resolving the next hop N does not match with the metric-type of Generic-Metric TLV of the AIGP attribute, then the BGP speaker may normalize cost of the path used for resolving the next hop. A policy may be used to provide the metric normalization.

7. Updates to Decision Procedure

This section follows the approach as laid out in [RFC7311] to select the best path when the route has AIGP attribute with Generic-Metric TLV. The domain that the router R belongs to, has enabled metric-types different from IGP cost. The following describes procedures in addition to general procedure described in section 4 of [RFC7311] .

When R receives a route T with next hop N and the AIGP attribute with Generic-Metric TLV, and the metric-type sub-field matches with the type of the metric of the path used for resolving the next hop N, the AIGP-enhanced interior cost should be computed as below.

Let m be the cost to reach the next hop N that IGP uses for its path computation as described in [RFC7311] .

If the type of the metric of the path used for resolving the next hop N does not match the metric-type sub-field of the Generic-Metric TLV, the cost of the path to reach next hop N may be normalized. The normalized metric value can be zero, maximum metric value or scaled up (multiple of a positive number).

Let m be the normalized value of the cost to reach the next hop N that IGP uses for its path computation as described in [RFC7311] .

The AIGP-enhanced interior cost computation as described below will be used in the decision process as described in [RFC7311] .

Let A be the value of the value of the metric-value sub-field of the Generic-Metric TLV.

The AIGP-enhanced interior cost will be $A+m$ as described in [RFC7311] .

A path with Generic-Metric TLV and a path with AIGP TLV cannot be compared. To enable end-to-end path selection based on intent, the with Generic-Metric TLV MUST be chosen over path with AIGP TLV. The implementation should allow a local policy to specify the preference.

A path with Generic-Metric TLV of metric-type 'a' cannot be compared with a path with Generic-Metric TLV of metric-type 'b'. The path with lower metric-type MUST be chosen as best between two paths with Generic-Metric TLV.

8. Use-case: Different Metrics across Domains

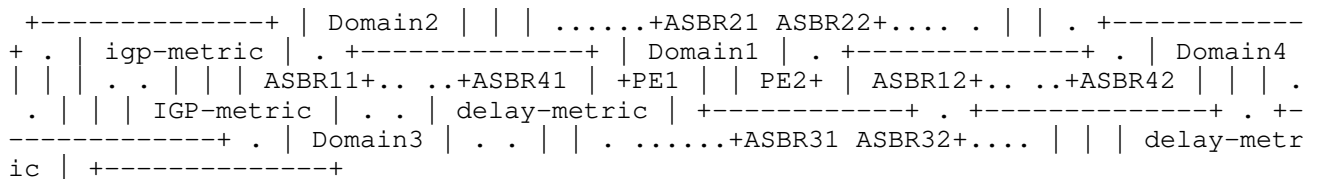


Figure 3: Different metric across network

Each domain is a separate Autonomous System. Within each domain, ASBR and PE form iBGP peering. The IGP within each domain uses domain specific metric. Domain3 and Domain4 use delay as the metric while Domain1 and Domain2 use IGP cost as the metric. ASBRs across domains form eBGP peering. The use-case is to find delay-based end-to-end path from Domain1 to Domain4.

This can be achieved by the advertising router to add the AIGP attribute with metric type 1 that represents delay metric. In the above network diagram, ASBR41 (and ASBR42) will advertise prefix PE2-loopback with Generic-Metric TLV with delay as metric-type. The metric-value sub-field of the Generic-Metric TLV will represent the cost to reach PE2's loopback end-point from the advertising router as they will do next hop self.

In Domain3, when ASRB32 advertises the prefix PE2-loopback within the local domain, it may add cost to the metric-value, the value representing the delay introduced by the DMZ link between ASRB32 to ASBR42. When ASRB31 advertises the prefix PE2-loopback, it will perform the following procedures.

1. Compute the delay d of the path to reach ASBR32 from which it has chosen the best path.
2. Add the above d value to the metric-value sub-field of the Generic-Metric TLV.

In Domain2 however, the local metric type IGP cost. The ASBR22 may follow the procedure similar to ASBR32 and add the delay value corresponding to the DMZ link between ASBR22 and ASBR41 before advertising the path internally in Domain2. When ASBR21 computes the AIGP-enhanced interior cost, as mentioned before, it may normalize the igp cost to reach ASBR22 and may add the normalized value to the delay-metric. In the above network example, the delay cost from ASBR21 to ASBR22 is negligible and hence delay-metric value will be unchanged.

The procedures for AIGP-enhanced interior cost computation at ASBR11 (and ASBR12) will follow DMZ delay computation procedure described above. PE1 will have two paths to reach PE2-loopback: P1 via ASBR11 (and domain2) and P2 via ASBR12 (and domain3), each having respective AIGP-enhanced interior cost representing end-to-end delay. The BGP decision process described in Section 7 will result in delay optimized end-to-end path for PE2-loopback on PE1 that can be used to resolve the service prefixes.

9. Deployment Considerations

It can be noted that a domain may normalize the metric-value of the metric-type of the path used to resolve next hop to the metric-type present in the Generic-Metric TLV. The idea is to propagate the cost of reaching the prefix through the domain while maintaining the metric-type chosen by the originating router and domain. The normalization of metric types to the one carried in the AIGP attribute can be done via policy. Definition of such policies and how they can be enforced is outside the scope of this document. In topologies where there is a common router between adjacent domains that do iBGP peering, the Border router can provide the normalization.

It is important to maintain the property of IGP cost to a destination decrease as one gets closer to the destination. The AIGP-enhanced interior cost should not be allowed to decrease through the metric normalization. When adjacent domains use different metric types, the ASBR that connects two domains is better suited to pass on the metric values by setting itself as next hop.

All routers of a domain MUST compute the AIGP-enhanced interior cost as described above to be used during decision process. Within a

domain, if one router R1 applies AIGP-enhanced interior cost while R2 does not, it may lead to routing loop unless some sort of tunnelling technology viz MPLS, SRv6, IP, etc. is adopted to reach the next hop. In a network where any tunnelling technology is used, one can incrementally deploy the Generic-Metric functionality. In a network without any tunnelling technology, it is recommended that all routers MUST support Generic-Metric based AIGP-enhanced interior cost computation.

The contiguity of the AIGP domain across multiple IGP or AS domains is important to maintain end-to-end path of a certain intent. A router that does not recognize Generic-Metric TLV, may add AIGP TLV and pass on the BGP route with just AIGP TLV. This results in AIGP attribute having both TLVs. The router making decision only on Generic-Metric TLV may chose sub-optimal paths.

In certain networks, routes may be redistributed between BGP and IGP, usually controlled via a policy. When a route is propagated across domains, a router should use AIGP metric-value of Generic-Metric TLV, optionally modified via the local policy as the IGP cost during route redistribution in to IGP. The local policy should apply metric normalization or translation based on metric-type of Generic-Metric TLV and the metric-type adopted in the IGP.

10. Contiguity Compliance

AIGP attribute is optional and non-transitive, however new TLV might not be interpreted and/or updated by routers along the path. For computing the end-to-end path based on an intent, it is essential to maintain contiguity of AIGP domain for the metric-type. The mechanism will be addressed in the future version of this document.

11. Backward Compatibility

When a BGP speaker receives an update with the AIGP attribute it may have Generic-Metric TLV. If the BGP speaker understands the AIGP attribute but does not understand the Generic-Metric TLV, it will process the AIGP attribute as per [RFC7311] . However when it needs to advertise the prefix to its peers it will pass on the AIGP attribute with all the TLVs including the unknown Generic-Metric TLV as per [RFC7311] . If a BGP speaker does not understand the Generic-Metric TLV, it may chose sub-optimal BGP path.

12. Security Considerations

This document does not introduce any new security considerations beyond those already specified in [RFC4271] , [RFC7311] .

13. IANA Considerations

IANA is requested to assign a code point for Generic Metric TLV. The metric-type field refers to the IGP metric-type registry defined in [I-D.ietf-lsr-flex-algo-bw-con]

14. Acknowledgements

The authors would like to thank John Scudder, Jeff Haas, Robert Raszuk, and Kaliraj Vairavakkalai for careful review and suggestions.

15. References

15.1. Normative References

[I-D.dskc-bess-bgp-car]

Rao, D., Agrawal, S., Filsfils, C., Talaulikar, K., Steinberg, D., Jalil, L., Su, Y., Decraene, B., Guichard, J., Patel, K., and H. Wang, "BGP Color-Aware Routing (CAR)", draft-dskc-bess-bgp-car-03 (work in progress), October 2021.

[I-D.dskc-bess-bgp-car-problem-statement]

Rao, D., Agrawal, S., Filsfils, C., Talaulikar, K., Decraene, B., Steinberg, D., Jalil, L., Guichard, J., Patel, K., and W. Henderickx, "BGP Color-Aware Routing Problem Statement", draft-dskc-bess-bgp-car-problem-statement-04 (work in progress), November 2021.

[I-D.hegde-spring-seamless-sr-architecture]

Hegde, S., Bowers, C., Xu, X., Gulko, A., Bogdanov, A., Uttaro, J., Jalil, L., Khaddam, M., and A. Alston, "Seamless Segment Routing Architecture", draft-hegde-spring-seamless-sr-architecture-00 (work in progress), February 2021.

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

Hegde, S., J. W. B. A., Shetty, R., Decraene, B., Psenak, P., and T. Li, "Flexible Algorithms: Bandwidth, Delay, Metrics and Constraints", draft-ietf-lsr-flex-algo-bw-con-01 (work in progress), July 2021.

[I-D.kaliraj-idr-bgp-classful-transport-planes]

Vairavakkalai, K., Venkataraman, N., Rajagopalan, B., Mishra, G., Khaddam, M., Xu, X., Szarecki, R. J., and D. J. Gowda, "BGP Classful Transport Planes", draft-kaliraj-idr-bgp-classful-transport-planes-13 (work in progress), January 2022.

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

15.2. Informative References

- [RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <<https://www.rfc-editor.org/info/rfc3630>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", RFC 4659, DOI 10.17487/RFC4659, September 2006, <<https://www.rfc-editor.org/info/rfc4659>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <<https://www.rfc-editor.org/info/rfc5305>>.

- [RFC7311] Mohapatra, P., Fernando, R., Rosen, E., and J. Uttaro, "The Accumulated IGP Metric Attribute for BGP", RFC 7311, DOI 10.17487/RFC7311, August 2014, <<https://www.rfc-editor.org/info/rfc7311>>.
- [RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <<https://www.rfc-editor.org/info/rfc7471>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.
- [RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <<https://www.rfc-editor.org/info/rfc8570>>.

Authors' Addresses

Srihari Sangli
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: ssangli@juniper.net

Shraddha Hegde
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: shraddha@juniper.net

Reshma Das
Juniper Networks Inc.
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: dreshma@juniper.net

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2022

W. Wang
A. Wang
China Telecom
H. Wang
Huawei Technologies
G. Mishra
Verizon Inc.
S. Zhuang
J. Dong
Huawei Technologies
September 30, 2021

Route Distinguisher Outbound Route Filter (RD-ORF) for BGP-4
draft-wang-idr-rd-orf-08

Abstract

This draft defines a new Outbound Route Filter (ORF) type, called the Route Distinguisher ORF (RD-ORF). The described RD-ORF mechanism is applicable when the VPN routes from different VRFs are exchanged via one shared BGP session(e.g. routers in a single-domain connect via Route Reflector).

Status of This Memo

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

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

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

This Internet-Draft will expire on April 3, 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.

Table of Contents

1. Introduction	2
2. Conventions used in this document	3
3. Terminology	4
4. Operation process of RD-ORF mechanism on sender	4
4.1. Intra-domain Scenarios and Solutions	4
4.1.1. Scenario-1 and Solution (Unique RD, One RT)	5
4.1.2. Scenario-2 and Solution (Unique RD, Multiple RTs)	6
4.1.3. Scenario-2 and Solution (Universal RD)	7
5. Operation process of RD-ORF mechanism on receiver	8
6. Withdraw of RD-ORF entries	8
7. RD-ORF Encoding	8
7.1. Source PE TLV	10
8. Security Considerations	10
9. IANA Considerations	10
10. Acknowledgement	11
11. Normative References	11
Authors' Addresses	12

1. Introduction

[I-D.wang-idr-vpn-routes-control-analysis] analysis the scenarios and necessities for VPN routes control in the shared BGP session. This draft analyzes the existing solutions and their limitations for these scenarios, proposes the new RD-ORF solution to meet the requirements that described in section 8 of [I-D.wang-idr-vpn-routes-control-analysis].

Now, there are several solutions can be used to alleviate these problem:

- o Route Target Constraint (RTC) as defined in [RFC4684]
- o Address Prefix ORF as defined in [RFC5292]
- o PE-CE edge peer Maximum Prefix
- o Configure the Maximum Prefix for each VRF on edge nodes

However, there are limitations to existing solutions:

1) Route Target Constraint

RTC can only filter the VPN routes from the uninterested VRFs, if the "trashing routes" come from the interested VRF, filter on RTs will erase all prefixes from this VRF.

2) Address Prefix ORF

Using Address Prefix ORF to filter VPN routes need to pre-configuration, but it is impossible to know which prefix may cause overflow in advance.

3) PE-CE edge peer Maximum Prefix

This mechanism can only protect the edge between PE-CE, it can't be deployed within PE that peered via RR. Depending solely on the edge protection is dangerous, because if only one of the edge points being comprised/error-configured/attacked, then all of PEs within domain are under risk.

4) Configure the Maximum Prefix for each VRF on edge nodes

When a VRF overflows, it stops the import of routes and log the extra VPN routes into its RIB. However, PEs still need to parse the BGP updates. These processes will cost CPU cycles and further burden the overflowing PE.

This draft defines a new ORF-type, called the Route Distinguisher ORF (RD-ORF). Using RD-ORF mechanism, VPN routes can be controlled based on RD. This mechanism is event-driven and does not need to be pre-configured. When a VRF of a router overflows, the router will find out the RD of excessive VPN routes in this VRF, and send a RD-ORF to its BGP peer that carries the RD. If a BGP speaker receives a RD-ORF entry from its BGP peer, it will filter the VPN routes it tends to send according to the entry.

RD-ORF is applicable when the VPN routes from different VRFs are exchanged via one shared BGP session.

2. Conventions used in this document

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

3. Terminology

The following terms are defined in this draft:

- o RD: Route Distinguisher, defined in [RFC4364]
- o ORF: Outbound Route Filter, defined in [RFC5291]
- o AFI: Address Family Identifier, defined in [RFC4760]
- o SAFI: Subsequent Address Family Identifier, defined in [RFC4760]
- o EVPN: BGP/MPLS Ethernet VPN, defined in [RFC7432]
- o RR: Router Reflector, provides a simple solution to the problem of IBGP full mesh connection in large-scale IBGP implementation.
- o VRF: Virtual Routing Forwarding, a virtual routing table based on VPN instance.

4. Operation process of RD-ORF mechanism on sender

The operation of RD-ORF mechanism on each device is independent, each of them makes a local judgement to determine whether it needs to send RD-ORF to its peers.

When the RD-ORF mechanism is triggered, the device must send an alarm information to network operators.

4.1. Intra-domain Scenarios and Solutions

For intra-AS VPN deployment, there are three scenarios:

- o RD is allocated per VPN/per PE, each VRF only import one RT(see Section 4.1).
- o RD is allocated per VPN/per PE. Multiple RTs are associated with such VPN routes, and be imported into different VRFs in other devices(see Section 4.2).
- o RD is allocated per VPN, each VRF imports one/multiple RTs(see Section 4.3).

The following sections will describe solutions to the above scenarios in detail.

4.1.1. Scenario-1 and Solution (Unique RD, One RT)

In this scenario, RD is allocated per VPN or per PE, each VRF only import one RT. We assume the network topology is shown in Figure 1.

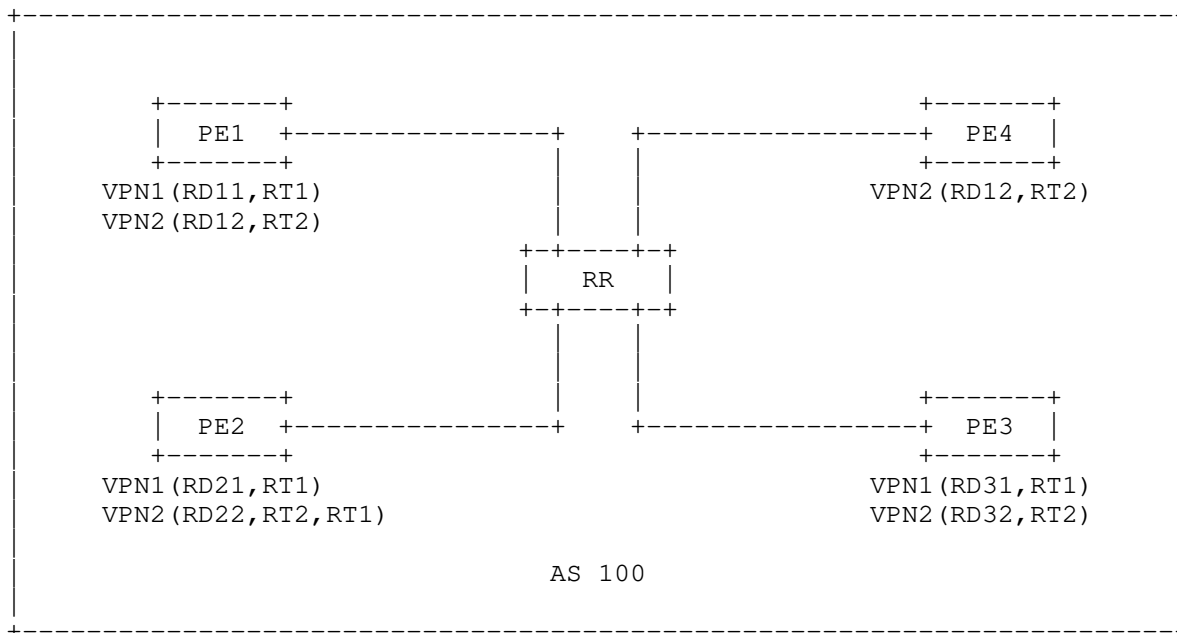


Figure 1 Network Topology of Scenario-1

When PE3 sends excessive VPN routes with RT1, while both PE1 and PE2 import VPN routes with RT1, the process of excessive VPN routes will influence performance of VRFs on PEs. PEs and RR should have some mechanisms to identify and control the advertisement of excessive VPN routes.

On PE1, each VRF has a set threshold, we assume it is 80% of Maximum Prefix of VRF. When the number of VPN1 VRF routing entries reaches the threshold, PE1 will start monitoring the RD carried by the received VPN routing entries. Once the number of VPN routing entries exceed the prefix limit, PE1 will calculate the RD and its source PE received the most times during this period, the result is RD31 from PE3, which is associated with RT1. Then, PE1 will locally discards the VPN routes carry RD31 which come from PE3 in VRF1.

Due to there is no other VRFs on it to import the VPN routes with RT1. after local processing, PE1 will generate a BGP ROUTE-REFRESH message contains a RD-ORF entry, and send to RR. RR will withdraw and stop to advertise such excessive VPN routes to PE1.

On PE2, the local processing is the same as PE1. Due to there has other VRF on it to import the VPN routes with RT1, PE2 triggers the RD-ORF message to RR(RD field is set to RD31) only when all the VRFs that import RT1 are overflowed.

4.1.2. Scenario-2 and Solution (Unique RD, Multiple RTs)

In this scenario, RD is allocated per VPN or per PE. Multiple RTs are associated with such VPN routes, and be imported into different VRFs in other devices. We assume the network topology is shown in Figure 2.

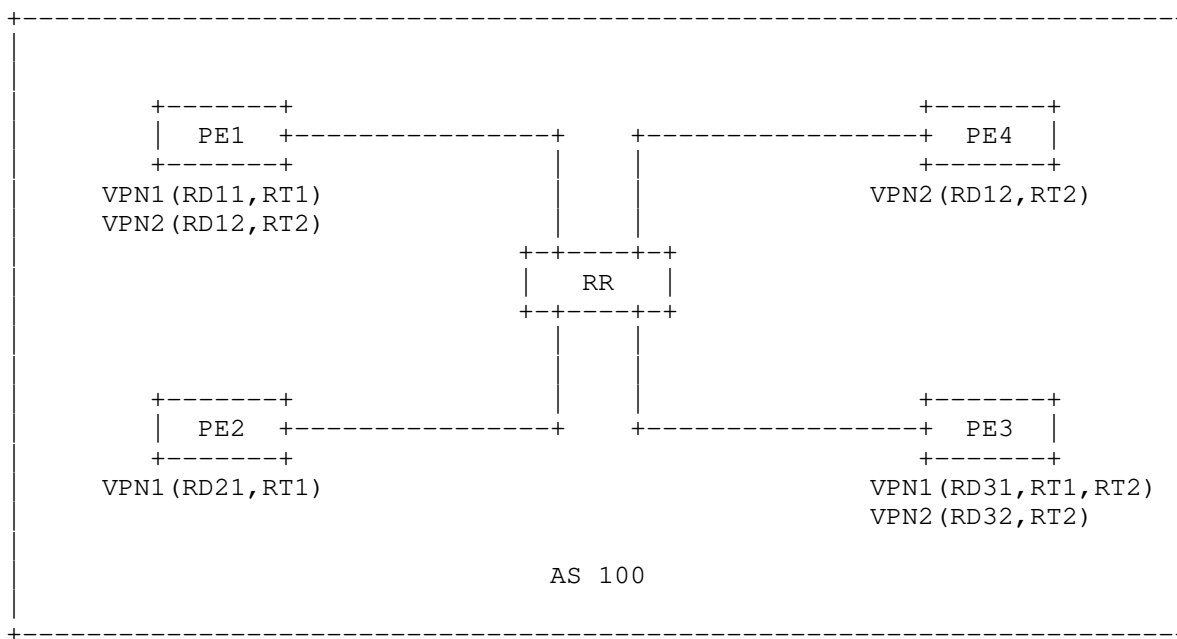


Figure 2 Network Topology of Scenario-2

When PE3 sends excessive VPN routes with RT1 and RT2, while both PE1 and PE2 import VPN routes with RT1, and PE1 also imports VPN routes with RT2, the process of excessive VPN routes will influence performance of VRF on PEs. PEs and RR should have some mechanisms to identify and control the advertisement of excessive VPN routes.

In this scenario, both VRF1 and VRF2 import VPN route carries RT2, which contains RD31.

On PE1, if it overflows, it will know that the RD of excessive VPN routes is RD31 during the local processing, which come from PE3 and associated with RT1 and RT2. There are different VRFs on PE1 import

the VPN routes respectively with RT1 and RT2. If PE1 trigger the RD-ORF message when VRF1 overflows, it cannot receive the VPN routes with RT2 from PE3. The local determination of the PE can be used to inhibit the PE from sending RD-ORF entries. PE1 will not trigger the RD-ORF message until all VPNs that import VPN routes with RD31 are overflowed. When RD-ORF mechanisms is triggered, PE1 will discard the overflowed VPN routes locally and send RD-ORF entry to RR, and RR withdraws and stops to advertise such excessive VPN routes to PE1.

On PE2, due to there is only one VRF imports VPN routes with RT1. If it overflows, it will trigger RD-ORF(RD31) mechanisms. RR will withdraw and stop to advertise such excessive VPN routes to PE2.

4.1.3. Scenario-2 and Solution (Universal RD)

In this scenario, RD is allocated per VPN. One/Multiple RTs are associated with such VPN routes, and be imported into different VRFs in other devices. We assume the network topology is shown in Figure 3.

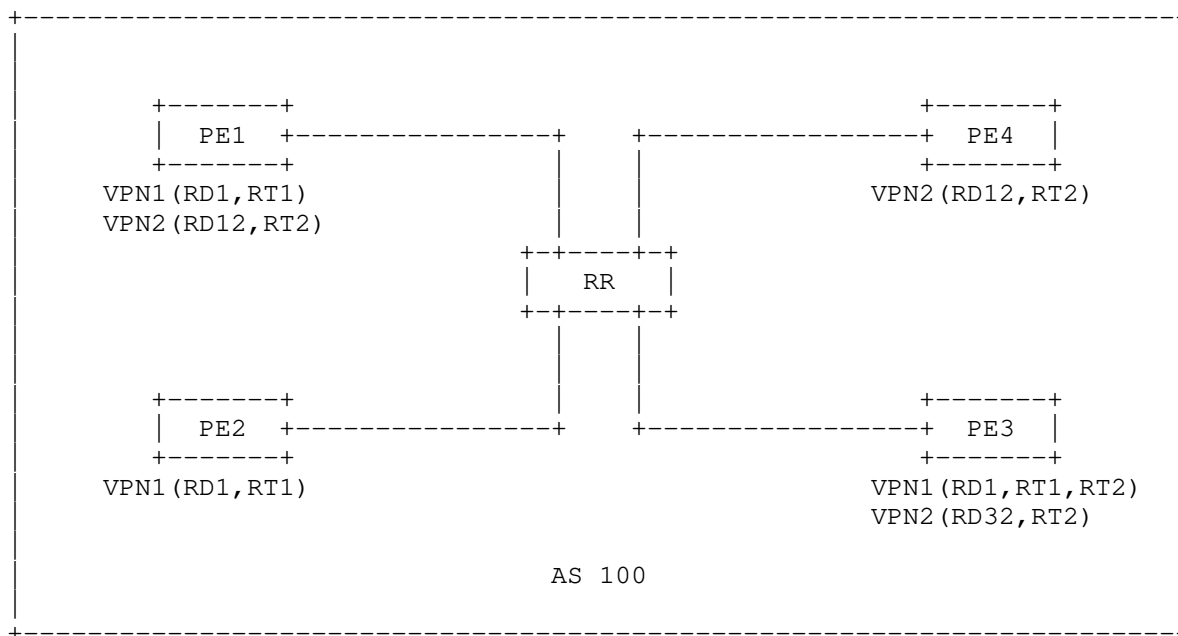


Figure 3 Network Topology of Scenario-3

When PE3 sends excessive VPN routes with RD1 and attached RT1 and RT2, while both PE1 and PE2 import VPN routes with RT1, the process of excessive VPN routes will influence performance of VRF on PEs.

PEs and RR should have some mechanisms to identify and control the advertisement of excessive VPN routes.

Based on previous principle, when PE2 overflows and PE1 not. PE2 triggers the RD-ORF message(RD1, comes from PE3). RR will withdraw and stop to advertise such excessive VPN routes to PE2. The communication between PE2 and PE1 for VPN1 will not be influenced.

5. Operation process of RD-ORF mechanism on receiver

The receiver of RD-ORF entries may be a RR or PE. As it receives the RD-ORF entries, it will check <AFI/SAFI, ORF-Type, Sequence, Route Distinguisher> to find if it already existed in its ORF-Policy table. If not, the receiver will add the RD-ORF entries into its ORF-Policy table; otherwise, the receiver will discard it. Before the receiver send a VPN route, it will check its ORF-Policy table whether there is a related RD-ORF entry or not. If not, the receiver will send this VPN route; otherwise, the receiver will stop sending that VPN route to its peer.

6. Withdraw of RD-ORF entries

When the RD-ORF mechanism is triggered, the alarm information will be generated and sent to the network operators. Operators should manually configure the network to resume normal operation. Due to devices can record the RD-ORF entries sent by each VRF, operators can find the entries needs to be withdrawn, and trigger the withdraw process as described in [RFC5291] manually. After returning to normal, the device sends withdraw ORF entries to its peers who have previously received ORF entries.

7. RD-ORF Encoding

In this section, we defined a new ORF type called Route Distinguisher Outbound Route Filter (RD-ORF). The ORF entries are carried in the BGP ROUTE-REFRESH message as defined in [RFC5291]. A BGP ROUTE-REFRESH message can carry one or more ORF entries. The ROUTE-REFRESH message which carries ORF entries contains the following fields:

- o AFI (2 octets)
- o SAFI (1 octet)
- o When-to-refresh (1 octet): the value is IMMEDIATE or DEFER
- o ORF Type (1 octet)
- o Length of ORF entries (2 octets)

A RD-ORF entry contains a common part and type-specific part. The common part is encoded as follows:

- o Action (2 bits): the value is ADD, REMOVE or REMOVE-ALL
- o Match (1 bit): the value is PERMIT or DENY
- o Reserved (5 bits)

RD-ORF also contains type-specific part. The encoding of the type-specific part is shown in Figure 4.

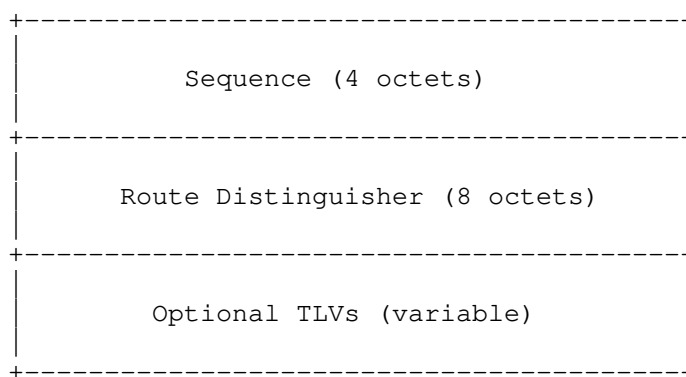


Figure 4: RD-ORF type-specific encoding

- o Sequence: identifying the order in which RD-ORF is generated
- o Route Distinguisher: distinguish the different user routes. The RD-ORF filters the VPN routes it tends to send based on Route Distinguisher.
- o Optional TLVs: carry the potential additional information to give the extensibility of the RD-ORF mechanism.

Note that if the Action component of an ORF entry specifies REMOVE-ALL, the ORF entry does not include the type-specific part. The Sequence can uniquely identifies an RD-ORF entry. All VRFs share the sequence field, and the corresponding sequence of RD-ORF sent by each VRF will be recorded on the device.

When the BGP ROUTE-REFRESH message carries RD-ORF entries, it must be set as follows:

- o The ORF-Type MUST be set to RD-ORF.

- o The AFI MUST be set to IPv4, IPv6, or Layer 2 VPN (L2VPN).
- o If the AFI is set to IPv4 or IPv6, the SAFI MUST be set to MPLS-labeled VPN address.
- o If the AFI is set to L2VPN, the SAFI MUST be set to BGP EVPN.
- o The Match field MUST be equal to DENY.

7.1. Source PE TLV

Source PE TLV is defined to identify the source of the VPN routes. Using source PE and RD to filter VPN routes together can achieve more refined route control. The source PE TLV contains the following types:

- o In single-domain or Option C cross-domain scenario, NEXT_HOP attribute is fixed during routing transmission, so it can be used as source address.

Type = 1, Length = 4 octets, value = NEXT_HOP.

Type = 2, Length = 16 octets, value = NEXT_HOP.
- o In Option B or Option AB cross-domain scenario, NEXT_HOP attribute may be changed by ASBRs and cannot be used as the source address. The originator can be traced by the Route Origin Community in BGP (as defined in Section 5 of [RFC4360]).

Type = 3, Length = 6 octets, value = the value field of Route Origin Community.

8. Security Considerations

A BGP speaker will maintain the RD-ORF entries in an ORF-Policy table, this behavior consumes its memory and compute resources. To avoid the excessive consumption of resources, [RFC5291] specifies that a BGP speaker can only accept ORF entries transmitted by its interested peers.

9. IANA Considerations

This document defines a new Outbound Route Filter type - Route Distinguisher Outbound Route Filter (RD-ORF). The code point is from the "BGP Outbound Route Filtering (ORF) Types". It is recommended to set the code point of RD-ORF to 66.

This document also define a RD-ORF TLV type under "Border Gateway Protocol (BGP) Parameters", three TLV types are defined:

Registry	Type	Meaning
IPv4 Source PE TLV	1	IPv4 address for source PE.
IPv6 Source PE TLV	2	IPv6 address for source PE.
ROC Source PE TLV	3	Route Origin Community for Source PE.

Figure 5: IANA Allocation for newly defined TLVs

10. Acknowledgement

Thanks Robert Raszuk, Jim Uttaro, Jakob Heitz, Jeff Tantsura, Rajiv Asati, John E Drake, Gert Doering, Shuanglong Chen, Enke Chen and Srihari Sangli for their valuable comments on this draft.

11. Normative References

- [I-D.ietf-bess-evpn-inter-subnet-forwarding]
Sajassi, A., Salam, S., Thoria, S., Drake, J. E., and J. Rabadan, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-15 (work in progress), July 2021.
- [I-D.wang-idr-vpn-routes-control-analysis]
Wang, A., Wang, W., Mishra, G. S., Wang, H., Zhuang, S., and J. Dong, "Analysis of VPN Routes Control in Shared BGP Session", draft-wang-idr-vpn-routes-control-analysis-04 (work in progress), September 2021.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <<https://www.rfc-editor.org/info/rfc4360>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.

- [RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <<https://www.rfc-editor.org/info/rfc4684>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC5291] Chen, E. and Y. Rekhter, "Outbound Route Filtering Capability for BGP-4", RFC 5291, DOI 10.17487/RFC5291, August 2008, <<https://www.rfc-editor.org/info/rfc5291>>.
- [RFC5292] Chen, E. and S. Sangli, "Address-Prefix-Based Outbound Route Filter for BGP-4", RFC 5292, DOI 10.17487/RFC5292, August 2008, <<https://www.rfc-editor.org/info/rfc5292>>.
- [RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

Authors' Addresses

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

Email: weiwang94@foxmail.com

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

Email: wangaj3@chinatelecom.cn

Haibo Wang
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: rainsword.wang@huawei.com

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

Shunwan Zhuang
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: zhuangshunwan@huawei.com

Jie Dong
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: jie.dong@huawei.com

IDR Working Group
Internet-Draft
Intended status: Informational
Expires: March 10, 2022

A. Wang
W. Wang
China Telecom
G. Mishra
Verizon Inc.
H. Wang
S. Zhuang
J. Dong
Huawei Technologies
September 6, 2021

Analysis of VPN Routes Control in Shared BGP Session
draft-wang-idr-vpn-routes-control-analysis-04

Abstract

This draft analyzes some scenarios and the necessities for VPN routes control in the shared BGP session, which can be the used as the base for the design of related solutions.

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 March 10, 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.

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
3. Terminology	2
4. Inter-AS VPN Option B/AB Scenario	3
5. Inter-AS VPN Option C Scenario	4
6. Intra-AS VPN RR Deployment Scenario	5
7. VPN Routes Shared on one PE	6
8. Requirements for the solutions	7
9. Security Considerations	8
10. IANA Considerations	8
11. Acknowledgement	8
12. Normative References	8
Authors' Addresses	8

1. Introduction

BGP Maximum Prefix feature [RFC4486] is often used at the network boundary to control the number of prefixes to be injected into the network. But for some scenarios when the VPN routes from several VRFs are advertised via one shared BGP session, there is lack of appropriate methods to control the flooding of VPN routes within one VRF to overwhelm the process of VPN routes in other VRFs. That is to say, the excessive VPN routes advertisement should be controlled individually for each VRF in such shared BGP session.

The following sections analyzes the scenarios that are necessary to such mechanism.

2. Conventions used in this document

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

3. Terminology

The following terms are defined in this draft:

- o RD: Route Distinguisher, defined in [RFC4364]

- o RR: Router Reflector, provides a simple solution to the problem of IBGP full mesh connection in large-scale IBGP implementation.
- o VRF: Virtual Routing Forwarding, a virtual routing table based on VPN instance.

4. Inter-AS VPN Option B/AB Scenario

For inter-AS VPN deployment option B/AB scenario, as described in Figure 1, there is one BGP session between ASBR1 and ASBR2, which is used to advertise the VPN routes from VPN1 and VPN2 VRF. Normally the operator will deploy the BGP maximum prefixes feature under different address families between the ASBR1 and ASBR2, but the threshold must be set very high to cope with the situation when all the VRFs in each family reach their VPN routes limit simultaneously. In case VPN routes in only one of VRF, for example VPN1 in PE3, advertises excess VPN routes(with RD set to RD31 and RT import/export set to RT1. Configurations on other PEs are similar) into the network, but VPN routes advertisement in other VRFs are in normal, the prefix bar set between the ASBRs will not take effect. Such excessive VPN routes will be advertised into the AS1, to PE1 and PE2 respectively.

PE1 in this example, provides the services for VPN2 at the same time. If it receives the excessive VPN routes for VPN1 from ASBR1, although such VPN routes have exceeded the limit within the VRF VPN1, it can't break the BGP session with ASBR1 directly, because the VPN prefix limit is to prevent a flood from errors or other issues but does not prevent the device from being overwhelmed and resources exhausted.

All it can do is to receive and process the excessive BGP updates continuously, parse the excessive VPN routes for VPN1 and drop it, extract the VPN routes for VPN2 and install it.

Doing so can certainly influence the performance of PE1 to serve the other VPN services on it, considering that there are hundreds of VRFs deployed on it.

PE1 should have the capability to control the advertisement of specified excessive VPN routes from its BGP peer. The ASBR should also have such capability.

The excessive VPN routes may carry just one RT(for example in VPN1 on PE3), or carry more than one RTs(for example in VPN2 on PE3). Such excessive VPN routes may be imported into one VRF(for example VPN1 on PE1) or more than one VRFs(for example both VPN2 and VPN3 import the VPN routes with RD32, which has attached RT2 and RT3 together when they are advertised)

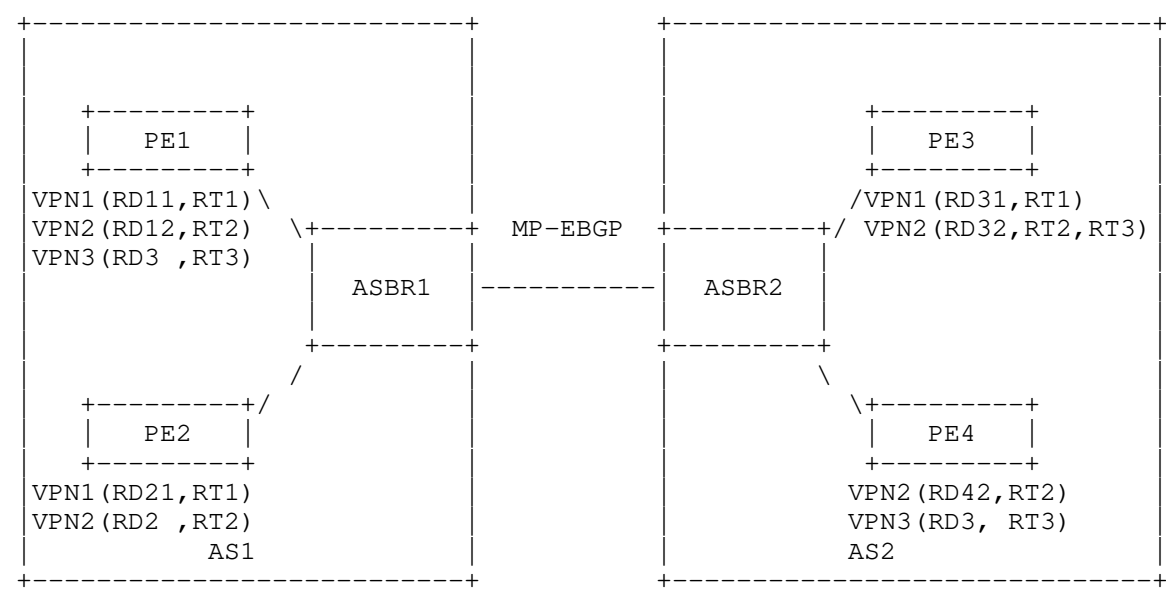


Figure 1: The Option B/Option AB cross-domain scenario

5. Inter-AS VPN Option C Scenario

For inter-AS VPN deployment option C scenario, as that described in Figure 2, there is one BGP session between RR1 and RR2, which is used to advertise the VPN routes from all the VRFs that located on the edge routers(PE1 and PE2). The BGP maximum prefix bar can't also prevent the excessive advertisement of VPN routes in one VRF, and such abnormal behavior in one VRF can certainly influence the performances of PEs to serve other normal VRFs.

PE and RR should all have some capabilities to control the specified excessive VPN routes to be advertised from its upstream BGP peer.

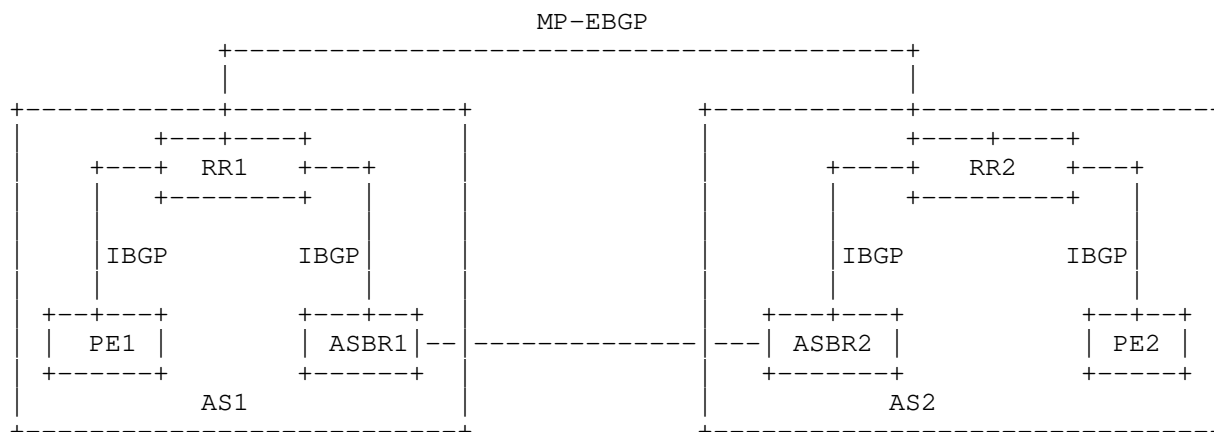


Figure 2: The Option C cross-domain scenario

6. Intra-AS VPN RR Deployment Scenario

For intra-AS VPN deployment, as depicted in Figure 3, if the RR is present, the above excess VPN routes advertisement churn can also occur. For example, if PE3 receives excessive VPN routes for VPN1 VRF (there may be several reasons for this to occur, for example, multiple CEs connect to PE3 advertising routes simultaneously causing a wave of routes, redistribution from VRF to VRF, or from GRT to VRF on PE3 etc.), it will advertise such excessive VPN routes to RR and then to PE1. The BGP session between RR and PE3, and the BGP session between RR and PE1 can't prevent this to occur.

The RD in each VPN may be allocated and unique for each VPN on each PE (as example VPN1 in Figure 3), or only unique for each VPN (as example VPN2 in Figure 3).

Each VPN may be associated with one or more RTs. The excessive VPN routes may have only one RT (for example, the excessive VPN routes from PE3 has the RD equal to RD31 and RT is set only to RT1)

When PE1 in this figure receives such excessive VPN routes, it can only process them, among the other normal BGP updates. This can certainly influence process of VPN routes for other normal services, the consequences on the receiving PE1 may be the one or more of the followings:

- a) PE1 can't process a given number of routes in time period X leading to dropping of routes

- b) Delayed processing that may result in an incomplete number of inputs to the BGP Best Path decision.
- c) L3VPN customers experiencing an incorrect VPN specification for some time period Y.
- d) The convergence of control plane processing impacts the traffic forwarding

PE and RR should all have some capabilities to control the specified excessive VPN routes to be advertised from its upstream BGP peer.

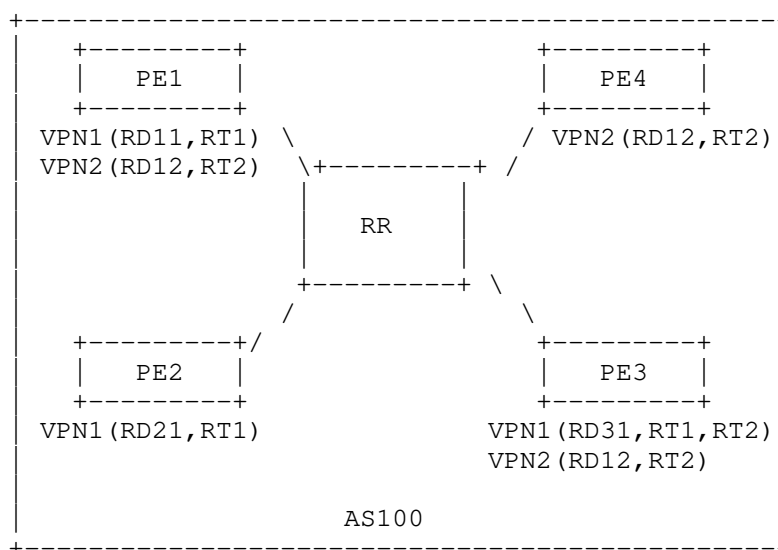


Figure 3: Intra-AS VPN RR deployment scenario

7. VPN Routes Shared on one PE

The scenarios described above are mainly in device level, that is to say, if the receiving PE has some mechanism to control the excess VPN routes advertisement from its BGP neighbor, the failure churn effect can be controlled then. But there are also situations that the granular control should be took place within the receiving PE itself.

Figure 4 below describes such scenario. There are four VRFs on PE, and three of them import the same VPN routes that carry route target RT3. Such deployment can occur in the inter-VRF communication scenario. If the threshold of VPN route-limit for these VRFs is set different, for example, are max-vpn-routes-vrf1, max-vpn-routes-vrf2, max-vpn-routes-vrf3, max-vpn-routes-vrf4 respectively, and these

values have the following order, as $\text{max-vpn-routes-vrf1} < \text{max-vpn-routes-vrf2} < \text{max-vpn-routes-vrf3} < \text{max-vpn-routes-vrf4}$.

If the VPN routes that associates with RT3 is overwhelming, the VRF1 will reach its maximum VPN threshold first. At such stage, the PE device can't send the control message to its BGP neighbor on behalf of all the VRFs on it, because other VRFs have still the desire to receive such VPN routes and have the capacities to store them.

In such situation, the PE device should have some mechanisms to control the distribution of global VPN routes to its individual VRF table. Only when all of VRFs on it don't want some VPN routes, then the PE device can send the VPN routes filter control message to its BGP neighbor (RR in this example).

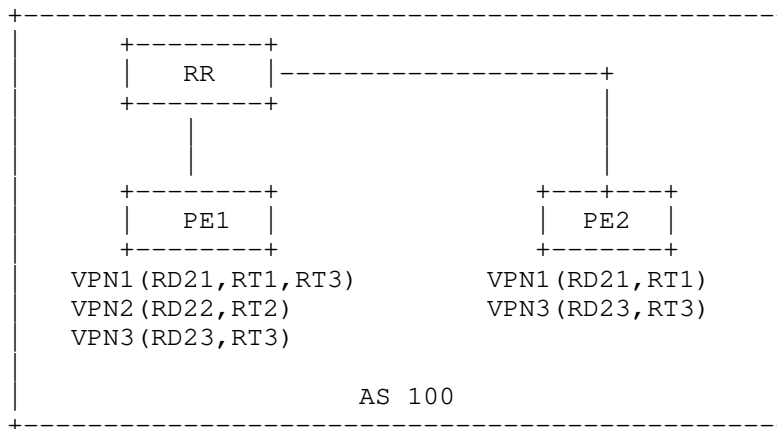


Figure 4: The scenario of several VRFs in a PE import VPN routes carries the same RT

.

8. Requirements for the solutions

Based on the above scenarios description, the potential solutions should meet the following requirements:

- a) The control message for the specified VPN routes should be triggered automatically upon the excessive VPN routes reach its limit.
- b) The control message should be sent only out the device when all the VRFs on it can't or don't want to process it, or the process of such excessive routes has exceed its own capability.

c) For RR devices, such control message should be only flooded to its upstream BGP neighbor when all its clients can't or don't want to process it, or the process of such excessive routes has exceed its own capability.

d) For ASBR devices, such control message should be only flooded to its upstream BGP neighbor when all its downstream BGP peers can't or don't want to process it, or the process of such excessive routes has exceed its own capability.

e) The trigger and removal of such control message should avoid the possible flapping of excessive VPN routes advertisement.

9. Security Considerations

TBD.

10. IANA Considerations

This document requires no IANA considerations.

11. Acknowledgement

Thanks Robert Raszuk, Jim Uttaro, Jakob Heitz, Shuanglong Chen, Enke Chen and Srihari Sangli for their valuable comments and discussions of scenarios described in this draft.

12. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4486] Chen, E. and V. Gillet, "Subcodes for BGP Cease Notification Message", RFC 4486, DOI 10.17487/RFC4486, April 2006, <<https://www.rfc-editor.org/info/rfc4486>>.

Authors' Addresses

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

Email: wangaj3@chinatelecom.cn

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

Email: wangw36@chinatelecom.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

Haibo Wang
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: rainsword.wang@huawei.com

Shunwan Zhuang
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: zhuangshunwan@huawei.com

Jie Dong
Huawei Technologies
Huawei Building, No.156 Beiqing Rd.
Beijing, Beijing 100095
China

Email: jie.dong@huawei.com

IDR Working Group
Internet-Draft
Intended status: Informational
Expires: 13 July 2022

C. Xie
C. Li
China Telecom
J. Dong
Z. Li
Huawei Technologies
9 January 2022

BGP-LS with Multi-topology for Segment Routing based Virtual Transport
Networks
draft-xie-idr-bgpls-sr-vtn-mt-04

Abstract

Enhanced VPN (VPN+) aims to provide enhanced VPN service to support some applications' needs of enhanced isolation and stringent performance requirements. VPN+ requires integration between the overlay VPN and the underlay network. A Virtual Transport Network (VTN) is a virtual underlay network which consists of a subset of the network topology and network resources allocated from the physical network. A VTN could be used as the underlay for one or a group of VPN+ services.

When Segment Routing is used as the data plane of VTNs, each VTN can be allocated with a group of Segment Identifiers (SIDs) to identify the topology and resource attributes of network segments in the VTN. The association between the network topology, the network resource attributes and the SR SIDs may need to be distributed to a centralized network controller. For network scenarios where each VTN can be associated with a unique logical network topology, this document describes a mechanism to distribute the information of SR based VTNs using BGP-LS with Multi-Topology.

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 13 July 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 SR VTN Topology Attribute	4
2.1. Intra-domain Topology Advertisement	4
2.2. Inter-Domain Topology Advertisement	5
3. Advertisement of SR VTN Resource Attribute	6
4. Scalability Considerations	7
5. Security Considerations	7
6. IANA Considerations	7
7. Acknowledgments	7
8. References	7
8.1. Normative References	7
8.2. Informative References	9
Authors' Addresses	10

1. Introduction

Enhanced VPN (VPN+) is an enhancement to VPN services to support the needs of new applications, particularly including 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. Thus these properties require integration between the overlay connectivity and the characteristics provided by the underlay 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 Transport Networks (VTNs) need to be created, each consists of a subset of the underlay network topology and a subset of network resources allocated from the underlay network to meet the requirement of one or a group of VPN+ services.

[I-D.ietf-spring-resource-aware-segments] introduces resource awareness to Segment Routing (SR) [RFC8402]. The resource-aware SIDs have additional semantics to identify the set of network resources available for the packet processing action associated with the SIDs. As described in [I-D.ietf-spring-sr-for-enhanced-vpn], the resource-aware segments can be used to build SR based VTNs with the required network topology and network resource attributes to support enhanced VPN services.

To allow the network controller and network nodes to perform VTN-specific explicit path computation and/or shortest path computation, the group of resource-aware SIDs allocated by network nodes to each VTN and the associated topology and resource attributes need to be distributed in the control plane. When a centralized network controller is used for VTN-specific path computation, especially when a VTN spans multiple IGP areas or multiple Autonomous Systems (ASes), BGP-LS is needed to advertise the VTN 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 VTNs.

In some network scenarios, each VTN can be associated with a unique logical network topology, [I-D.ietf-lsr-isis-sr-vtn-mt] describes an IGP mechanism to advertise the association between the topology, resource attributes and the SR SIDs for each VTN. This document describes a mechanism to distribute the information of SR based VTNs to the network controller using BGP-LS with Multi-Topology.

2. Advertisement of SR VTN Topology Attribute

[I-D.ietf-lsr-isis-sr-vtn-mt] describes the IS-IS Multi-topology based mechanisms to distribute the topology attributes of SR based VTNs. This section describes the corresponding BGP-LS mechanism to distribute both the intra-domain and inter-domain topology attributes of SR based VTNs.

2.1. Intra-domain 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.

[I-D.ietf-idr-bgp-ls-segment-routing-ext] defines the BGP-LS extensions to carry the segment routing information using TLVs of BGP-LS Attribute. When Multi-Topology 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-bgp-ls-srv6-ext] defines the BGP-LS extensions to advertise SRv6 segments along with their functions and attributes. When Multi-Topology 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 may be elaborated in a future version.

2.2. Inter-Domain Topology Advertisement

[I-D.ietf-idr-bgppls-segment-routing-epe] and [I-D.ietf-idr-bgppls-srv6-ext] defines the BGP-LS extensions for advertisement of BGP inter-domain topology information and the BGP Egress Peering Segment Identifiers. Such information could be used by a network controller for the computation and instantiation of inter-AS traffic engineering SR paths.

In some network scenarios, there are needs to create VTNs which span multiple ASes. The inter-domain VTNs 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 SR based VTNs, it is necessary to advertise the topology and resource attribute of each VTN and the associated BGP Peering SIDs on the inter-domain links.

Depending on the requirement of inter-domain VTNs, different mechanism can be used on the inter-domain connection:

- * 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 VTNs 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 several pieces, each of which can be considered as a virtual member link. A VTN is associated with one of the physical or virtual member links. In both cases, different BGP Peer-Adj-SIDs or SRv6 End.X SID SHOULD be allocated to each underlying physical or virtual member link, the association between the BGP Peer Adj-SID/End.X SID and the identifier of the VTN SHOULD be advertised by the ASBR.

- * 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 multi-domain VTN, while some other BGP sessions are used for another multi-domain VTN. In this case, different BGP Peer Node SIDs are allocated to each BGP session and are advertised using the mechanism in [I-D.ietf-idr-bgppls-segment-routing-epe] and [I-D.ietf-idr-bgppls-srv6-ext], the association between the BGP Peer Node SIDs and the identifier of the VTN SHOULD be advertised by the ASBR.
- * At the AS-level topology, different multi-domain VTNs may have different inter-domain connectivity. Different BGP Peer Set SIDs MAY be allocated to represent the groups of BGP peers which can be used for load-balancing in each multi-domain VTN.

When MT-ID is used consistently in multiple ASes covered by a VTN, the topology-specific BGP peering SIDs can be advertised with the MT-ID carried in the corresponding Link NLRI. This can be achieved with the existing mechanisms as defined in [RFC7752][I-D.ietf-idr-bgppls-segment-routing-epe] and [I-D.ietf-idr-bgppls-srv6-ext].

In network scenarios where consistent usage of MT-ID among multiple domains can not be expected, a global-significant VTN-ID needs to be introduced to define the inter-domain topologies. Within each domain, the MT based mechanism could be reused for intra-domain topology advertisement. The detailed mechanism is specified in [I-D.dong-idr-bgppls-sr-enhanced-vpn].

3. Advertisement of SR VTN Resource Attribute

[I-D.ietf-lsr-isis-sr-vtn-mt] specifies the mechanism to advertise the resource information associated with each VTN. This section describes the corresponding BGP-LS mechanisms.

The information of the network resources associated with a VTN can be specified by carrying the TE Link attribute TLVs in BGP-LS Attribute [RFC7752], with the associated MT-ID carried in the corresponding Link NLRI.

When Maximum Link Bandwidth sub-TLV is carried in the BGP-LS attribute associated with the Link NLRI of a VTN, it indicates the amount of link bandwidth resource allocated to the corresponding VTN on the link. The bandwidth allocated to a VTN can be exclusive for traffic in the corresponding VTN. The advertisement of other TE attributes in BGP-LS for each VTN is for further study.

4. Scalability Considerations

The mechanism described in this document requires that each VTN mapped to an independent topology, and for the inter-domain VTNs, the MT-IDs used in each involved domain need to be consistent. Reusing MT-IDs as the identifier of VTN can avoid introducing new identifiers in the control plane, while it also has some limitations. For example, when multiple VTNs shares the same topology, each VTN still need to be identified using different MT-IDs in the control plane, thus independent path computation needs be executed for each VTN. The number of VTNs supported in a network may be dependent on the number of topologies supported, which is related to the control plane overhead. The mechanism described in this document is applicable to network scenarios where the number of required VTN is relatively small. A detailed analysis about the VTN scalability and the possible optimizations for supporting a large number of VTNs is described in [I-D.dong-teas-enhanced-vpn-vtn-scalability].

5. 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.

6. IANA Considerations

This document does not request any IANA actions.

7. Acknowledgments

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

8. References

8.1. Normative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H.,
and M. Chen, "Border Gateway Protocol - Link State (BGP-
LS) Extensions for Segment Routing", Work in Progress,
Internet-Draft, draft-ietf-idr-bgp-ls-segment-routing-ext-
18, 15 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgp-ls-segment-routing-ext-18.txt>>.

- [I-D.ietf-idr-bgppls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering", Work in Progress, Internet-Draft, draft-ietf-idr-bgppls-segment-routing-epe-19, 16 May 2019, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgppls-segment-routing-epe-19.txt>>.
- [I-D.ietf-idr-bgppls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Chen, M., Bernier, D., and B. Decraene, "BGP Link State Extensions for SRv6", Work in Progress, Internet-Draft, draft-ietf-idr-bgppls-srv6-ext-09, 10 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-idr-bgppls-srv6-ext-09.txt>>.
- [I-D.ietf-idr-rfc7752bis]
Talaulikar, K., "Distribution of Link-State and Traffic Engineering Information Using BGP", Work in Progress, Internet-Draft, draft-ietf-idr-rfc7752bis-10, 10 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-idr-rfc7752bis-10.txt>>.
- [I-D.ietf-spring-resource-aware-segments]
Dong, J., Bryant, S., Miyasaka, T., Zhu, Y., Qin, F., Li, Z., and F. Clad, "Introducing Resource Awareness to SR Segments", Work in Progress, Internet-Draft, draft-ietf-spring-resource-aware-segments-03, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-spring-resource-aware-segments-03.txt>>.
- [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, Internet-Draft, draft-ietf-spring-sr-for-enhanced-vpn-01, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-spring-sr-for-enhanced-vpn-01.txt>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC5029] Vasseur, JP. and S. Previdi, "Definition of an IS-IS Link Attribute Sub-TLV", RFC 5029, DOI 10.17487/RFC5029, September 2007, <<https://www.rfc-editor.org/info/rfc5029>>.

- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

8.2. Informative References

- [I-D.dong-idr-bgppls-sr-enhanced-vpn]
Dong, J., Hu, Z., Li, Z., Tang, X., and R. Pang, "BGP-LS Extensions for Segment Routing based Enhanced VPN", Work in Progress, Internet-Draft, draft-dong-idr-bgppls-sr-enhanced-vpn-03, 22 February 2021, <<https://www.ietf.org/archive/id/draft-dong-idr-bgppls-sr-enhanced-vpn-03.txt>>.
- [I-D.dong-lsr-sr-enhanced-vpn]
Dong, J., Hu, Z., Li, Z., Tang, X., Pang, R., JooHeon, L., and S. Bryant, "IGP Extensions for Scalable Segment Routing based Enhanced VPN", Work in Progress, Internet-Draft, draft-dong-lsr-sr-enhanced-vpn-06, 11 July 2021, <<https://www.ietf.org/archive/id/draft-dong-lsr-sr-enhanced-vpn-06.txt>>.
- [I-D.dong-teas-enhanced-vpn-vtn-scalability]
Dong, J., Li, Z., Gong, L., Yang, G., Guichard, J. N., Mishra, G., and F. Qin, "Scalability Considerations for Enhanced VPN (VPN+)", Work in Progress, Internet-Draft, draft-dong-teas-enhanced-vpn-vtn-scalability-04, 25 October 2021, <<https://www.ietf.org/archive/id/draft-dong-teas-enhanced-vpn-vtn-scalability-04.txt>>.
- [I-D.ietf-lsr-isis-sr-vtn-mt]
Xie, C., Ma, C., Dong, J., and Z. Li, "Using IS-IS Multi-Topology (MT) for Segment Routing based Virtual Transport Network", Work in Progress, Internet-Draft, draft-ietf-lsr-isis-sr-vtn-mt-01, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-lsr-isis-sr-vtn-mt-01.txt>>.
- [I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Segment Routing over

IPv6 Dataplane", Work in Progress, Internet-Draft, draft-ietf-lsr-isis-srv6-extensions-18, 20 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lsr-isis-srv6-extensions-18.txt>>.

[I-D.ietf-teas-enhanced-vpn]

Dong, J., Bryant, S., Li, Z., Miyasaka, T., and Y. Lee, "A Framework for Enhanced Virtual Private Network (VPN+) Services", Work in Progress, Internet-Draft, draft-ietf-teas-enhanced-vpn-09, 25 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-teas-enhanced-vpn-09.txt>>.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <<https://www.rfc-editor.org/info/rfc5120>>.

[RFC8667] Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", RFC 8667, DOI 10.17487/RFC8667, December 2019, <<https://www.rfc-editor.org/info/rfc8667>>.

Authors' Addresses

Chongfeng Xie
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing
102209
China

Email: xiechf@chinatelecom.cn

Cong Li
China Telecom
China Telecom Beijing Information Science & Technology, Beiqijia
Beijing
102209
China

Email: licong@chinatelecom.cn

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing
100095
China

Email: jie.dong@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Campus, No. 156 Beiqing Road
Beijing
100095
China

Email: lizhenbin@huawei.com

idr
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2022

Z. Zhang
J. Haas
Juniper Networks
K. Patel
Arrcus
4 March 2022

Extended Communities Derived from Route Targets
draft-zzhang-idr-rt-derived-community-02

Abstract

This document specifies a way to derive an Extended Community from a Route Target and describes some example use cases.

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	2
2. Use Cases	3
2.1. EVPN EVI-RT Extended Community	3
2.2. Leaf Discovery with Controller Signaled BGP-MVPN	3
2.3. Translated Route-target Extended Communities in [I-D.ietf-idr-legacy-rtc]	4
3. Security Considerations	4
4. IANA Assignments	4
5. References	5
5.1. Normative References	5
5.2. Informative References	5
Authors' Addresses	5

1. Introduction

Consider a VPN with 10 PEs. A Route Target (say RT1) [RFC4360] is configured for the VPN and all PEs will import VPN routes with RT1 attached. The RT is an Extended Community (say EC1), with its sub-type being 0x02. While RT1 and EC1 have the same encoding, typically when we mention a Route Target, its property of being able to control the route propagation and importation is implied. When we just mention an Extended Community, that property is not implied.

Now consider that another BGP route needs to be imported by some but not all those PEs. The route could be of any SAFI/type (does not need to be a VPN prefix), but it needs to be associated with the VPN on those importing PEs. The exact meaning of "association" here does not matter, but the key is that those PEs need to know that the route is related to that VPN.

To control the propagation to and importation by those PEs, a different Route Target (say RT3) is attached to the route. For those PE to associate the route with the VPN, an Extended Community (say EC2) is attached. Even though RT1/EC1 is already associated with the VPN, EC2 needs to be different from RT1/EC1, because if EC1 was used, the route would be propagated to and imported by all the 10 PEs. EC2 cannot be the same as RT3 either, because there could be other routes to be propagated to and imported by those same set of PEs yet those other routes are not related to the VPN.

While EC2 can be any Extended Community (that is not a RT) configured on the originating and receiving PEs to map it to the VPN, it is convenient if EC2 is derived from the RT1/EC1, e.g. the sub-type of RT1/EC1 is changed to a new known value while everything else remains the same. We call this a Route Target derived Extended Community, or RT-derived EC. A new sub-type is assigned specifically for this purpose (see IANA considerations).

This document only specifies a way to derive an Extended Community from a Route Target Extended Community using IANA-assigned Extended Community sub-types (or Extended Community Type in case of IPv6-Address-Specific Extended Community). Any protocol/feature that can take advantages of the convenience of generic derivation may use them, or not use them at its own discretion, and how they are used is outside the scope of this document.

2. Use Cases

The following are a few examples of use cases. To reiterate, these are example scenarios where generic RT-derived ECs could be used (when the routes to which they are attached provide enough context). It is not the intention of this document to mandate that it must be used.

2.1. EVPN EVI-RT Extended Community

Section 9.5 "EVI-RT Extended Community" of [I-D.ietf-bess-evpn-igmp-mld-proxy] describes a situation similar to the above. As a solution, four EVPN specific EVI-RT ECs are defined, each mapping to a type of Route Target for the corresponding EVPN instance.

As a theoretical alternative, a RT-derived EC described in this document could be used instead - just derive a generic EC from the EVI RT. Note that this document does not attempt to change the existing procedures in [I-D.ietf-bess-evpn-igmp-mld-proxy], but merely use it for illustration purposes.

2.2. Leaf Discovery with Controller Signaled BGP-MVPN

In Section 2 "Alternative to BGP-MVPN" of [I-D.ietf-bess-bgp-multicast-controller], BGP MCAST-TREE SAFI signaling can be used for a controller to program multicast forwarding state in VRFs of ingress/egress PEs, instead of relying on distributed BGP-MVPN signaling. For the controller to learn egress PEs of a VPN customer multicast tree (so that it can build/find a corresponding provider tunnel), egress PEs signal leaf information to the controller via Leaf Auto-Discovery routes. The routes carry a

Route Target for the controller (so that only the controller receives them), and an EC derived from the VPN's Route Target (so that the controller knows which VPN they are for).

2.3. Translated Route-target Extended Communities in [I-D.ietf-idr-legacy-rtc]

In Section 3.1 of [I-D.ietf-idr-legacy-rtc], a similar mechanism is described, as quoted below:

"The translation of the IRTs is necessary in order to refrain from importing "route-filter" VRF routes into VPN VRFs that would import the same route-targets. The translation of the IRTS is done as follows. For a given IRT, the equivalent translated RT (TRT) is constructed by means of swapping the value of the high-order octet of the Type field for the IRT (as defined in [RFC4360])."

3. Security Considerations

This document specifies a way to derive an Extended Community from a Route Target Extended Community and does not specify how derived Extended Communities are used. As a result, this document does not need security considerations. Any potential security concerns need be addressed by documents that specify the actual usage.

4. IANA Assignments

IANA has assign a new sub-type "RT-derived-EC" with value 0x15 in the following registries:

- * Transitive Two-Octet AS-Specific Extended Community Sub-Types
- * Transitive Four-Octet AS-Specific Extended Community Sub-Types
- * Transitive IPv4-Address-Specific Extended Community Sub-Types
- * Non-Transitive Opaque Extended Community Sub-Types
- * EVPN Extended Community Sub-Types

IANA has also assigned a new type "RT-derived-EC" with value 0x0015 in the following registry:

- * Transitive IPv6-Address-Specific Extended Community Types

If and when additional Extended Community types are defined with a Route Target sub-type, the "RT-derived-EC" sub-type may also be registered for those new types, preferably with the same value.

5. References

5.1. Normative References

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

5.2. Informative References

[I-D.ietf-bess-evpn-igmp-mld-proxy]
Sajassi, A., Thoria, S., Mishra, M., Drake, J., and W. Lin, "IGMP and MLD Proxy for EVPN", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-igmp-mld-proxy-19, 4 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-bess-evpn-igmp-mld-proxy-19.txt>>.

[I-D.ietf-bess-bgp-multicast-controller]
Zhang, Z., Raszuk, R., Pacella, D., and A. Gulko, "Controller Based BGP Multicast Signaling", Work in Progress, Internet-Draft, draft-ietf-bess-bgp-multicast-controller-07, 12 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-bess-bgp-multicast-controller-07.txt>>.

[I-D.ietf-idr-legacy-rtc]
Mohapatra, P., Sreekantiah, A., Patel, K., Pithawala, B., and A. Lo, "Automatic Route Target Filtering for legacy PEs", Work in Progress, Internet-Draft, draft-ietf-idr-legacy-rtc-08, 12 September 2017, <<https://www.ietf.org/archive/id/draft-ietf-idr-legacy-rtc-08.txt>>.

Authors' Addresses

Zhaohui Zhang
Juniper Networks
Email: zzhang@juniper.net

Jeff Haas
Juniper Networks
Email: jhaas@juniper.net

Keyur Patel
Arrcus
Email: keyur@arrcus.com

idr
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2022

Z. Zhang
Juniper Networks
July 07, 2021

MPLS Label Stacks in Tunnel Encapsulation Attribute
draft-zzhang-idr-tunnel-encapsulation-label-stack-00

Abstract

[RFC9012] defines an MPLS Label Stack sub-TLV for Tunnel Encapsulation Attribute, and specifies that it is to be pushed BEFORE other labels. This document clarifies the use case for that, and defines a new Tunnel Label Stack sub-TLV for a label stack to be pushed AFTER other labels (e.g., the label embedded in the NLRI for a labeled address family, and/or the stack in an MPLS Label Stack sub-TLV).

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 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

[RFC9012] defines an MPLS Label Stack sub-TLV for Tunnel Encapsulation Attribute and specifies that:

"If a packet is to be sent through the tunnel identified in a particular TLV, and if that TLV contains an MPLS Label Stack sub-TLV, then the label stack appearing in the sub-TLV MUST be pushed onto the packet before any other labels are pushed onto the packet."

Specifically, the label stack in the sub-TLV is to be pushed BEFORE any other labels are pushed. This may sound counter-intuitive, in that if a label stack (e.g. for Segment Routing) is to be used to steer traffic to the tunnel endpoint, the stack should be pushed AFTER other labels (e.g. the label embedded in the NLRI).

This document clarifies that it is NOT for steering traffic to but for steering AFTER the tunnel endpoint. Consider the following use case:

```

      controller
      .
      .
      .
site1 --- PE1 ----- PE2 --- site2
```

Two sites are connected to two PEs respectively, and traffic steering is desired within each site not just among the PEs. While PE2 could push the label stack used for steering within site2, there may be situations where PE2 is not a device capable of pushing a large label stack so PE1 is tasked with pushing the label stack that is used after the tunnel end point PE2, within site2.

2. Tunnel Label Stack sub-TLV

Notice that in the above example, the route update that PE1 receives could be from the controller instead of PE2. The controller may want to steer traffic both within site2 and between PE1 and PE2, yet RFC9012 currently does not specify how to signal the label stack used to reach the tunnel end point.

To be able to signal that, this document defines a new Tunnel Label Stack sub-TLV. It has exactly the same syntax as the existing MPLS Label Stack sub-TLV, but with a semantics that it is pushed AFTER other labels are pushed.

3. Security Considerations

This document does not introduce any new security issues besides what is already discussed in RFC9012.

4. IANA Assignments

IANA is requested to assign a new sub-TLV type for "Tunnel Label Stack" from "BGP Tunnel Encapsulation Attribute Sub-TLVs" registry, in the 0~127 range.

5. Acknowledgements

The authors thank Eric Rosen and John Scudder for their valuable comments and suggestions.

6. Normative References

[RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <<https://www.rfc-editor.org/info/rfc9012>>.

Author's Address

Zhaohui Zhang
Juniper Networks

Email: zzhang@juniper.net