

SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 9, 2018

F. Clad, Ed.
C. Filsfils
P. Camarillo
Cisco Systems, Inc.
D. Bernier
Bell Canada
B. Decraene
Orange
B. Peirens
Proximus
C. Yadlapalli
AT&T
X. Xu
Huawei
S. Salsano
Universita di Roma "Tor Vergata"
A. Abdelsalam
Gran Sasso Science Institute
G. Dawra
Cisco Systems, Inc.
October 6, 2017

Segment Routing for Service Chaining
draft-clad-spring-segment-routing-service-chaining-00

Abstract

This document defines data plane functionality required to implement service segments and achieve service chaining with MPLS and IPv6, as described in the Segment Routing architecture.

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 9, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Terminology	4
3. Classification and steering	4
4. Services	5
4.1. SR-aware services	5
4.2. SR-unaware services	6
5. SR proxy behaviors	6
5.1. Static SR proxy	9
5.1.1. SR-MPLS pseudocode	10
5.1.2. SRv6 pseudocode	11
5.2. Dynamic SR proxy	13
5.2.1. SR-MPLS pseudocode	14
5.2.2. SRv6 pseudocode	15
5.3. Shared memory SR proxy	15
5.4. Masquerading SR proxy	15
5.4.1. SRv6 masquerading proxy pseudocode - End.AM	17
5.4.2. Variant 1: NAT	17
5.4.3. Variant 2: Caching	17
6. Illustrations	18
7. Metadata	20
7.1. MPLS data plane	20
7.2. IPv6 - SRH TLV objects	20
7.3. IPv6 - SRH tag	20
8. Implementation status	20
9. Relationship with RFC 7665	21
10. IANA Considerations	21
11. Security Considerations	22
12. Acknowledgements	22
13. Contributors	22
14. References	22
14.1. Normative References	22

14.2. Informative References	22
Authors' Addresses	23

1. Introduction

Segment Routing (SR) is an architecture based on the source routing paradigm that seeks the right balance between distributed intelligence and centralized programmability. SR can be used with an MPLS or an IPv6 data plane to steer packets through an ordered list of instructions, called segments. These segments may encode simple routing instructions for forwarding packets along a specific network path, or rich behaviors to support use-cases such as service chaining.

In the context of service chaining, each service, running either on a physical appliance or in a virtual environment, is associated with a segment, which can then be used in a segment list to steer packets through the service. Such service segments may be combined together in a segment list to achieve service chaining, but also with other types of segments as defined in [I-D.ietf-spring-segment-routing]. SR thus provides a fully integrated solution for service chaining, overlay and underlay optimization. Furthermore, the IPv6 dataplane natively supports metadata transportation as part of the SR information attached to the packets.

This document describes how SR enables service chaining in a simple and scalable manner, from the segment association to the service up to the traffic classification and steering into the service chain. Several SR proxy behaviors are also defined to support SR service chaining through legacy, SR-unaware, services in various circumstances.

The definition of control plane components, such as segment discovery and SR policy configuration, is outside the scope of this data plane document. These aspects will be defined in a dedicated document.

Familiarity with the following IETF documents is assumed:

- o Segment Routing Architecture [I-D.ietf-spring-segment-routing]
- o Segment Routing with MPLS data plane
[I-D.ietf-spring-segment-routing-mpls]
- o Segment Routing Header [I-D.ietf-6man-segment-routing-header]
- o SRv6 Network Programming
[I-D.filsfils-spring-srv6-network-programming]

2. Terminology

SR-aware service: Service fully capable of processing SR traffic

SR-unaware service: Service unable to process SR traffic or behaving incorrectly for such traffic

SR proxy: Proxy handling the SR processing on behalf of an SR-unaware service

Service Segment: Segment associated with a service, either directly or via an SR proxy

SR SC policy: SR policy, as defined in [I-D.filsfils-spring-segment-routing-policy], that includes at least one Service Segment. An SR SC policy may also contain other types of segments, such as VPN or TE segments.

SR policy head-end: SR node that classifies and steers traffic into an SR policy.

3. Classification and steering

Classification and steering mechanisms are defined in section 12 of [I-D.filsfils-spring-segment-routing-policy] and are independent from the purpose of the SR policy. From a headend perspective, there is no difference whether a policy contains IGP, BGP, peering, VPN and service segments, or any combination of these.

As documented in the above reference, traffic is classified when entering an SR domain. The SR policy head-end may, depending on its capabilities, classify the packets on a per-destination basis, via simple FIB entries, or apply more complex policy routing rules requiring to look deeper into the packet. These rules are expected to support basic policy routing such as 5-tuple matching. In addition, the IPv6 SRH tag field defined in [I-D.ietf-6man-segment-routing-header] can be used to identify and classify packets sharing the same set of properties. Classified traffic is then steered into the appropriate SR policy, which is associated with a weighted set of segment lists.

SR traffic can be re-classified by an SR endpoint along the original SR policy (e.g., DPI service) or a transit node intercepting the traffic. This node is the head-end of a new SR policy that is imposed onto the packet, either as a stack of MPLS labels or as an IPv6 and SRH encapsulation.

4. Services

A service may be a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element. Unless otherwise stated, this document does not make any assumption on the type or execution environment of a service.

SR enables service chaining by assigning a segment identifier, or SID, to each service and sequencing these service SIDs in a segment list. A service SID may be of local significance or directly reachable from anywhere in the routing domain. The latter is realized with SR-MPLS by assigning a SID from the global label block ([I-D.ietf-spring-segment-routing-mpls]), or with SRv6 by advertising the SID locator in the routing protocol ([I-D.filsfils-spring-srv6-network-programming]).

This document categorizes services in two types, depending on whether they are able to behave properly in the presence of SR information or not. These are respectively named SR-aware and SR-unaware services. An SR-aware service can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the service own behavior is not hindered due to the presence of SR information. For example, an SR-aware firewall filtering SRv6 traffic based on its final destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header. Any service that does not meet these criteria is considered as SR-unaware.

4.1. SR-aware services

An SR-aware service is associated with a locally instantiated service segment, which is used to steer traffic through it.

If the service is configured to intercept all the packets passing through the appliance, the underlying routing system only has to implement a default SR endpoint behavior (SR-MPLS node segment or SRv6 End function), and the corresponding SID will be used to steer traffic through the service.

If the service requires the packets to be directed to a specific virtual interface, networking queue or process, a dedicated SR behavior may be required to steer the packets to the appropriate location. The definition of such service-specific functions is out of the scope of this document.

An SRv6-aware service may also retrieve, store or modify information in the SRH TLVs.

4.2. SR-unaware services

An SR-unaware service is not able to process the SR information in the traffic that it receives. It may either drop the traffic or take erroneous decisions due to the unrecognized routing information. In order to include such services in an SR SC policy, it is thus required to remove the SR information before the service receives the packet, or to alter it in such a way that the service can correctly process the packet.

In this document, we define the concept of an SR proxy as an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a service. The SR proxy can run as a separate process on the service appliance, on a virtual switch or router on the compute node or on a remote host. In this document, we only assume that the proxy is connected to the service via a layer-2 link.

An SR-unaware service is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the service. Section 5 describes several SR proxy behaviors to handle the SR information under various circumstances.

5. SR proxy behaviors

This section describes several SR proxy behaviors designed to enable SR service chaining through SR-unaware services. A system implementing one of these functions may handle the SR processing on behalf of an SR-unaware service and allows the service to properly process the traffic that is steered through it.

A service may be located at any hop in an SR policy, including the last segment. However, the SR proxy behaviors defined in this section are dedicated to supporting SR-unaware services at intermediate hops in the segment list. In case an SR-unaware service is at the last segment, it is sufficient to ensure that the SR information is ignored (IPv6 routing extension header with Segments Left equal to 0) or removed before the packet reaches the service (MPLS PHP, SRv6 End.D or PSP).

As illustrated on Figure 1, the generic behavior of an SR proxy has two parts. The first part is in charge of passing traffic from the network to the service. It intercepts the SR traffic destined for the service via a locally instantiated service segment, modifies it in such a way that it appears as non-SR traffic to the service, then

sends it out on a given interface, IFACE-OUT, connected to the service. The second part receives the traffic coming back from the service on IFACE-IN, restores the SR information and forwards it according to the next segment in the list. Unless otherwise stated IFACE-OUT and IFACE-IN can represent the same interface.

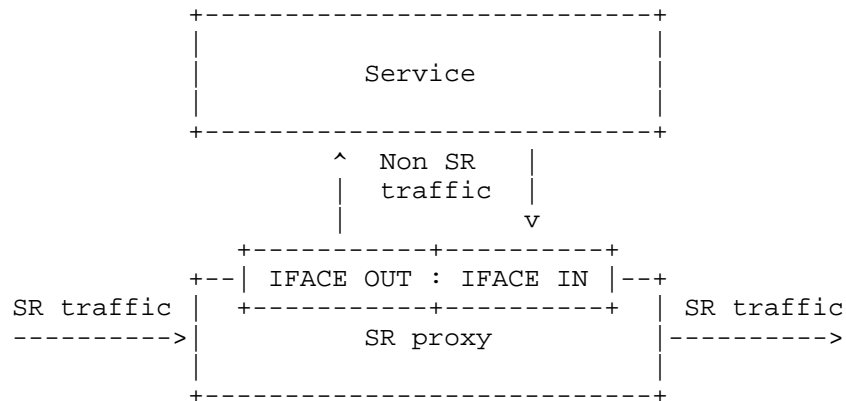


Figure 1: Generic SR proxy

In the next subsections, the following SR proxy mechanisms are defined:

- o Static proxy
- o Dynamic proxy
- o Shared-memory proxy
- o Masquerading proxy

Each mechanism has its own characteristics and constraints, which are summarized in the below table. It is up to the operator to select the best one based on the proxy node capabilities, the service behavior and the traffic type. It is also possible to use different proxy mechanisms within the same service chain.

		S t a t i c	D y n a m i c	S h a r e d m e m .	M a s q u e r a d i n g
SR flavors	SR-MPLS	Y	Y	Y	-
	SRv6 insertion	P	P	P	Y
	SRv6 encapsulation	Y	Y	Y	-
Inner header	Ethernet	Y	Y	Y	-
	IPv4	Y	Y	Y	-
	IPv6	Y	Y	Y	-
Chain agnostic configuration		N	N	Y	Y
Transparent to chain changes		N	Y	Y	Y
Service support	DA modification	Y	Y	Y	NAT
	Payload modification	Y	Y	Y	Y
	Packet generation	Y	Y	cache	cache
	Packet deletion	Y	Y	Y	Y
	Transport endpoint	Y	Y	cache	cache

Figure 2: SR proxy summary

Note: The use of a shared memory proxy requires both the service and the proxy to be running on the same node.

5.1. Static SR proxy

The static proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an MPLS label stack or an IPv6 header on top of an inner packet, which can be Ethernet, IPv4 or IPv6.

A static SR proxy segment is associated with the following mandatory parameters:

- o INNER-TYPE: Inner packet type
- o S-ADDR: Ethernet or IP address of the service (only for inner type IPv4 and IPv6)
- o IFACE-OUT: Local interface for sending traffic towards the service
- o IFACE-IN: Local interface receiving the traffic coming back from the service
- o CACHE: SR information to be attached on the traffic coming back from the service

A static SR proxy segment is thus defined for a specific service, inner packet type and cached SR information. It is also bound to a pair of directed interfaces on the proxy. These may be both directions of a single interface, or opposite directions of two different interfaces. The latter is recommended in case the service is to be used as part of a bi-directional SR SC policy. If the proxy and the service both support 802.1Q, IFACE-OUT and IFACE-IN can also represent sub-interfaces.

The first part of this behavior is triggered when the proxy node receives a packet whose active segment matches a segment associated with the static proxy behavior. It removes the SR information from the packet then sends it on a specific interface towards the associated service. This SR information corresponds to the full label stack for SR-MPLS or to the encapsulation IPv6 header with any attached extension header in the case of SRv6.

The second part is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy attaches to the incoming traffic the cached SR information associated with the SR proxy segment. If the proxy segment uses the SR-MPLS data plane, CACHE contains a stack of labels to be pushed on top the packets. With the SRv6 data plane, CACHE is defined as a source address, an active segment and an optional SRH (tag, segments

left, segment list and metadata). The proxy encapsulates the packets with an IPv6 header that has the source address, the active segment as destination address and the SRH as a routing extension header. After the SR information has been attached, the packets are forwarded according to the active segment, which is represented by the top MPLS label or the IPv6 Destination Address.

In this scenario, there are no restrictions on the operations that can be performed by the service on the stream of packets. It may operate at all protocol layers, terminate transport layer connections, generate new packets and initiate transport layer connections. This behavior may also be used to integrate an IPv4-only service into an SRv6 policy. However, a static SR proxy segment can be used in only one service chain at a time. As opposed to most other segment types, a static SR proxy segment is bound to a unique list of segments, which represents a directed SR SC policy. This is due to the cached SR information being defined in the segment configuration. This limitation only prevents multiple segment lists from using the same static SR proxy segment at the same time, but a single segment list can be shared by any number of traffic flows. Besides, since the returning traffic from the service is re-classified based on the incoming interface, an interface can be used as receiving interface (IFACE-IN) only for a single SR proxy segment at a time. In the case of a bi-directional SR SC policy, a different SR proxy segment and receiving interface are required for the return direction.

5.1.1.1. SR-MPLS pseudocode

5.1.1.1.1. Static proxy for inner type Ethernet - MPLS L2 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS L2 static proxy segment, a node N does:

1. IF payload type is Ethernet THEN
2. Pop all labels
3. Forward the exposed frame on IFACE-OUT
4. ELSE
5. Drop the packet

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

1. Push labels in CACHE on top of the frame Ethernet header
2. Lookup the top label and proceed accordingly

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

5.1.1.2. Static proxy for inner type IPv4 - MPLS IPv4 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv4 static proxy segment, a node N does:

1. IF payload type is IPv4 THEN
2. Pop all labels
3. Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5. Drop the packet

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

1. Push labels in CACHE on top of the packet IPv4 header
2. Decrement inner TTL and update checksum
3. Lookup the top label and proceed accordingly

5.1.1.3. Static proxy for inner type IPv6 - MPLS IPv6 static proxy segment

Upon receiving an MPLS packet with top label L, where L is an MPLS IPv6 static proxy segment, a node N does:

1. IF payload type is IPv6 THEN
2. Pop all labels
3. Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5. Drop the packet

Upon receiving a non link-local IPv6 packet on IFACE-IN, a node N does:

1. Push labels in CACHE on top of the packet IPv6 header
2. Decrement inner Hop Limit
3. Lookup the top label and proceed accordingly

5.1.2. SRv6 pseudocode

5.1.2.1. Static proxy for inner type Ethernet - End.AS2

Upon receiving an IPv6 packet destined for S, where S is an End.AS2 SID, a node N does:

1. IF ENH == 59 THEN ;; Ref1
2. Remove the (outer) IPv6 header and its extension headers
3. Forward the exposed frame on IFACE-OUT
4. ELSE
5. Drop the packet

Ref1: 59 refers to "no next header" as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving on IFACE-IN an Ethernet frame with a destination address different than the interface address, a node N does:

1. IF CACHE.SRH THEN ;; Ref2
2. Push CACHE.SRH on top of the existing Ethernet header
3. Set NH value of the pushed SRH to 59
4. Push outer IPv6 header with SA, DA and traffic class from CACHE
5. Set outer payload length and flow label
6. Set NH value to 43 if an SRH was added, or 59 otherwise
7. Lookup outer DA in appropriate table and proceed accordingly

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

The receiving interface must be configured in promiscuous mode in order to accept those Ethernet frames.

5.1.2.2. Static proxy for inner type IPv4 - End.AS4

Upon receiving an IPv6 packet destined for S, where S is an End.AS4 SID, a node N does:

1. IF ENH == 4 THEN ;; Ref1
2. Remove the (outer) IPv6 header and its extension headers
3. Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5. Drop the packet

Ref1: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non link-local IPv4 packet on IFACE-IN, a node N does:

1. IF CACHE.SRH THEN ;; Ref2
2. Push CACHE.SRH on top of the existing IPv4 header
3. Set NH value of the pushed SRH to 4
4. Push outer IPv6 header with SA, DA and traffic class from CACHE
5. Set outer payload length and flow label
6. Set NH value to 43 if an SRH was added, or 4 otherwise
7. Decrement inner TTL and update checksum
8. Lookup outer DA in appropriate table and proceed accordingly

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

5.1.2.3. Static proxy for inner type IPv6 - End.AS6

Upon receiving an IPv6 packet destined for S, where S is an End.AS6 SID, a node N does:

1. IF ENH == 41 THEN ;; Ref1
2. Remove the (outer) IPv6 header and its extension headers
3. Forward the exposed packet on IFACE-OUT towards S-ADDR
4. ELSE
5. Drop the packet

Ref1: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.

Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N does:

1. IF CACHE.SRH THEN ;; Ref2
2. Push CACHE.SRH on top of the existing IPv6 header
3. Set NH value of the pushed SRH to 41
4. Push outer IPv6 header with SA, DA and traffic class from CACHE
5. Set outer payload length and flow label
6. Set NH value to 43 if an SRH was added, or 41 otherwise
7. Decrement inner Hop Limit
8. Lookup outer DA in appropriate table and proceed accordingly

Ref2: CACHE.SRH represents the SRH defined in CACHE, if any, for the static SR proxy segment associated with IFACE-IN.

5.2. Dynamic SR proxy

The dynamic proxy is an improvement over the static proxy that dynamically learns the SR information before removing it from the incoming traffic. The same information can then be re-attached to the traffic returning from the service. As opposed to the static SR proxy, no CACHE information needs to be configured. Instead, the

dynamic SR proxy relies on a local caching mechanism on the node instantiating this segment. Therefore, a dynamic proxy segment cannot be the last segment in an SR SC policy. As mentioned at the beginning of Section 5, a different SR behavior should be used if the service is meant to be the final destination of an SR SC policy.

Upon receiving a packet whose active segment matches a dynamic SR proxy function, the proxy node pops the top MPLS label or applies the SRv6 End behavior, then compares the updated SR information with the cache entry for the current segment. If the cache is empty or different, it is updated with the new SR information. The SR information is then removed and the inner packet is sent towards the service.

The cache entry is not mapped to any particular packet, but instead to an SR SC policy identified by the receiving interface (IFACE-IN). Any non-link-local IP packet or non-local Ethernet frame received on that interface will be re-encapsulated with the cached headers as described in Section 5.1. The service may thus drop, modify or generate new packets without affecting the proxy.

5.2.1. SR-MPLS pseudocode

The static proxy SR-MPLS pseudocode is augmented by inserting the following instructions between lines 1 and 2.

```
1.  IF top label S bit is 0 THEN
2.      Pop top label
3.      IF C(IFACE-IN) different from remaining labels THEN ;; Ref1
4.          Copy all remaining labels into C(IFACE-IN)      ;; Ref2
5.  ELSE
6.      Drop the packet
```

Ref1: A TTL margin can be configured for the top label stack entry to prevent constant cache updates when multiple equal-cost paths with different hop counts are used towards the SR proxy node. In that case, a TTL difference smaller than the configured margin should not trigger a cache update (provided that the labels are the same).

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the MPLS label stack, and drop the packet otherwise.

5.2.2. SRv6 pseudocode

The static proxy SRv6 pseudocode is augmented by inserting the following instructions between lines 1 and 2.

```
1.  IF NH=SRH & SL > 0 THEN
2.      Decrement SL and update the IPv6 DA with SRH[SL]
3.      IF C(IFACE-IN) different from IPv6 encaps THEN      ;; Ref1
4.          Copy the IPv6 encaps into C(IFACE-IN)          ;; Ref2
5.  ELSE
6.      Drop the packet
```

Ref1: "IPv6 encaps" represents the IPv6 header and any attached extension header.

Ref2: C(IFACE-IN) represents the cache entry associated to the dynamic SR proxy segment. It is identified with IFACE-IN in order to efficiently retrieve the right SR information when a packet arrives on this interface.

In addition, the inbound policy should check that C(IFACE-IN) has been defined before attempting to restore the IPv6 encapsulation, and drop the packet otherwise.

5.3. Shared memory SR proxy

The shared memory proxy is an SR endpoint behavior for processing SR-MPLS or SRv6 encapsulated traffic on behalf of an SR-unaware service. This proxy behavior leverages a shared-memory interface with the service in order to hide the SR information from an SR-unaware service while keeping it attached to the packet. We assume in this case that the proxy and the service are running on the same compute node. A typical scenario is an SR-capable vrouter running on a container host and forwarding traffic to virtual services isolated within their respective container.

More details will be added in a future revision of this document.

5.4. Masquerading SR proxy

The masquerading proxy is an SR endpoint behavior for processing SRv6 traffic on behalf of an SR-unaware service. This proxy thus receives SR traffic that is formed of an IPv6 header and an SRH on top of an inner payload. The masquerading behavior is independent from the inner payload type. Hence, the inner payload can be of any type but it is usually expected to be a transport layer packet, such as TCP or UDP.

A masquerading SR proxy segment is associated with the following mandatory parameters:

- o S-ADDR: Ethernet or IPv6 address of the service
- o IFACE-OUT: Local interface for sending traffic towards the service
- o IFACE-IN: Local interface receiving the traffic coming back from the service

A masquerading SR proxy segment is thus defined for a specific service and bound to a pair of directed interfaces or sub-interfaces on the proxy. As opposed to the static and dynamic SR proxies, a masquerading segment can be present at the same time in any number of SR SC policies and the same interfaces can be bound to multiple masquerading proxy segments. The only restriction is that a masquerading proxy segment cannot be the last segment in an SR SC policy.

The first part of the masquerading behavior is triggered when the proxy node receives an IPv6 packet whose Destination Address matches a masquerading proxy segment. The proxy inspects the IPv6 extension headers and substitutes the Destination Address with the last segment in the SRH attached to the IPv6 header, which represents the final destination of the IPv6 packet. The packet is then sent out towards the service.

The service receives an IPv6 packet whose source and destination addresses are respectively the original source and final destination. It does not attempt to inspect the SRH, as RFC2460 specifies that routing extension headers are not examined or processed by transit nodes. Instead, the service simply forwards the packet based on its current Destination Address. In this scenario, we assume that the service can only inspect, drop or perform limited changes to the packets. For example, Intrusion Detection Systems, Deep Packet Inspectors and non-NAT Firewalls are among the services that can be supported by a masquerading SR proxy. Variants of the masquerading behavior are defined in Section 5.4.2 and Section 5.4.3 to support a wider range of services.

The second part of the masquerading behavior, also called de-masquerading, is an inbound policy attached to the proxy interface receiving the traffic returning from the service, IFACE-IN. This policy inspects the incoming traffic and triggers a regular SRv6 endpoint processing (End) on any IPv6 packet that contains an SRH. This processing occurs before any lookup on the packet Destination Address is performed and it is sufficient to restore the right active segment as the Destination Address of the IPv6 packet.

5.4.1. SRv6 masquerading proxy pseudocode - End.AM

Masquerading: Upon receiving a packet destined for S, where S is an End.AM SID, a node N processes it as follows.

1. IF NH=SRH & SL > 0 THEN
2. Update the IPv6 DA with SRH[0]
3. Forward the packet on IFACE-OUT
4. ELSE
5. Drop the packet

De-masquerading: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

1. IF NH=SRH & SL > 0 THEN
2. Decrement SL
3. Update the IPv6 DA with SRH[SL] ;; Ref1
4. Lookup DA in appropriate table and proceed accordingly

Ref2: This pseudocode can be augmented to support the Penultimate Segment Popping (PSP) endpoint flavor. The exact pseudocode modification are provided in [I-D.filsfils-spring-srv6-network-programming].

5.4.2. Variant 1: NAT

Services modifying the destination address in the packets they process, such as NATs, can be supported by a masquerading proxy with the following modification to the de-masquerading pseudocode.

De-masquerading - NAT: Upon receiving a non-link-local IPv6 packet on IFACE-IN, a node N processes it as follows.

1. IF NH=SRH & SL > 0 THEN
2. Update SRH[0] with the IPv6 DA
3. Decrement SL
4. Update the IPv6 DA with SRH[SL]
5. Lookup DA in appropriate table and proceed accordingly

5.4.3. Variant 2: Caching

Services generating packets or acting as endpoints for transport connections can be supported by adding a dynamic caching mechanism similar to the one described in Section 5.2.

More details will be added in a future revision of this document.

6. Illustrations

We consider the network represented in Figure 3 where:

- o A and B are two end hosts using IPv4
- o B advertises the prefix 20.0.0.0/8
- o 1 to 6 are physical or virtual routers supporting IPv6 and segment routing
- o S1 is an SR-aware firewall service
- o S2 is an SR-unaware IPS service

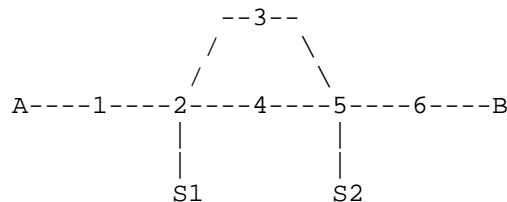


Figure 3: Network with services

All links are configured with an IGP weight of 10 except link 2-3 that is set to 20.

We assume that the path 2-3-5 has a lower latency than 2-4-5.

Nodes 1 to 6 each advertise in the IGP an IPv6 prefix $Ck::/64$, where k represents the node identifier.

Nodes 1 to 6 are each configured with an SRv6 End segment $Ck::/128$, where k represents the node identifier.

Node S1 is configured with an SRv6 SID $CF1::/128$ such that packets arriving at S1 with the Destination Address $CF1::$ are processed by the service. This SID is either advertised by S1, if it participates in the IGP, or by node 2 on behalf of S1.

Node 5 is also configured with an SRv6 dynamic proxy segments (End.AD) $C5::AD:F2$ for S2.

Node 6 is also configured with an SRv6 End.DX4 segment $C6::D4:B$ decapsulating the SRv6 and sending the inner IPv4 packets towards D.

Via BGP signaling or an SDN controller, node 1 is programmed with a route 20.0.0.0/8 via C6::D4:B and a color/community requiring low latency and services S1 and S2.

Node 1 either locally computes the path to the egress node or delegates the computation to a PCE. As a result, the SRv6 encapsulation policy < CF1::, C3::, C5::AD:F2, C6::D4:B > is associated with the route 20.0.0.0/8 on node 1.

Upon receiving a packet P from node A and destined to 20.20.20.20, node 1 finds the above table entry and pushes an outer IPv6 header with (SA = C1::, DA = CF1::, NH = SRH) followed by an SRH (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 3; NH = IPv4). Node 1 then forwards the packet to the first destination address CF1::.

Node 2 forwards P along the shortest path to S1, based on the IPv6 destination address CF1::.

When S1 receives the packet, it identifies a locally instantiated SID and applies the firewall filtering rules. If the packet is not dropped, the SL value is decremented and the DA is updated to the next segment C3::. S1 then sends back to node 2 the packet P with (SA = C1::, DA = C3::, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 2; NH = IPv4).

Node 2 forwards P along the shortest path to node 3, based on the IPv6 destination address C3::.

When 3 receives the packet, 3 matches the DA in its local SID table and finds the bound End function. It thus decrements the SL value and updates the DA to the next segment: C5::AD:F2. Node 3 then forwards packet P with (SA = C1::, DA = C5::AD:F2, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 1; NH = IPv4) towards node 5.

When 5 receives the packet, 5 matches the DA in its local SID table and finds the bound function End.AD(S2). It thus performs the End function (decrement SL and update DA), caches and removes the outer IPv6 header and the SRH, then forwards the inner IPv4 packet towards S2.

S2 receives a regular IPv4 packet headed to 20.20.20.20. It applies the IPS rules and forwards the packet back to node 5.

When 5 receives the packet on the returning interface (IFACE-IN) for S2, 5 retrieves the corresponding cache entry and pushes the updated IPv6 header and SRH. It then forwards P with (SA = C1::, DA = C6::D4:B, NH = SRH) (C6::D4:B, C5::AD:F2, C3::, CF1::; SL = 0; NH = IPv4) to node 6.

When 6 receives the packet, 6 matches the DA in its local SID table and finds the bound function End.DX4. It thus removes the outer IPv6 header and forwards the inner IPv4 packet to node B.

7. Metadata

7.1. MPLS data plane

The MPLS data plane does not provide any native mechanism to attach metadata to a packet.

Workarounds to carry metadata in an SR-MPLS context will be discussed in a future version of this document.

7.2. IPv6 - SRH TLV objects

The IPv6 SRH TLV objects are designed to carry all sorts of metadata. In particular, [I-D.ietf-6man-segment-routing-header] defines the NSH carrier TLV as a container for NSH metadata.

TLV objects can be imposed by the ingress edge router that steers the traffic into the SR SC policy.

An SR-aware service may impose, modify or remove any TLV object attached to the first SRH, either by directly modifying the packet headers or via a control channel between the service and its forwarding plane.

An SR-aware service that re-classifies the traffic and steers it into a new SR SC policy (e.g. DPI) may attach any TLV object to the new SRH.

Metadata imposition and handling will be further discussed in a future version of this document.

7.3. IPv6 - SRH tag

The SRH tag identifies a packet as part of a group or class of packets [I-D.ietf-6man-segment-routing-header].

In a service chaining context, this field can be used as a simple man's metadata to encode additional information in the SRH.

8. Implementation status

The static SR proxy is available for SR-MPLS and SRv6 on various Cisco hardware and software platforms. Furthermore, the following proxies are available on open-source software.

		VPP	Linux
M P L S	Static proxy	Available	In progress
	Dynamic proxy	In progress	In progress
	Shared memory proxy	In progress	In progress
S R v 6	Static proxy	Available	In progress
	Dynamic proxy - Inner type Ethernet	In progress	In progress
	Dynamic proxy - Inner type IPv4	Available	Available
	Dynamic proxy - Inner type IPv6	Available	Available
	Shared memory proxy	In progress	In progress
	Masquerading proxy	Available	Available
	Masquerading proxy - NAT variant	In progress	In progress
	Masquerading proxy - Cache variant	In progress	In progress

Open-source implementation status table

9. Relationship with RFC 7665

The Segment Routing solution addresses a wider problem that covers both topological and service chaining policies. The topological and service instructions can be either deployed in isolation or in combination. SR has thus a wider applicability than the architecture defined in [RFC7665]. Furthermore, the inherent property of SR is a stateless network fabric. In SR, there is no state within the fabric to recognize a flow and associate it with a policy. State is only present at the ingress edge of the SR domain, where the policy is encoded into the packets. This is completely different from NSH that relies on state configured at every hop of the service chain.

Hence, there is no linkage between this document and [RFC7665].

10. IANA Considerations

This document has no actions for IANA.

11. Security Considerations

The security requirements and mechanisms described in [I-D.ietf-spring-segment-routing] and [I-D.ietf-6man-segment-routing-header] also apply to this document. Additional considerations will be discussed in future versions of the document.

12. Acknowledgements

TBD.

13. Contributors

Jisu Bhattacharya substantially contributed to the content of this document.

14. References

14.1. Normative References

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S.,
and R. Shakir, "Segment Routing Architecture", draft-ietf-
spring-segment-routing-12 (work in progress), June 2017.

14.2. Informative References

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Raza, K., Liste, J., Clad,
F., Lin, S., bogdanov@google.com, b., Horneffer, M.,
Steinberg, D., Decraene, B., and S. Litkowski, "Segment
Routing Policy for Traffic Engineering", draft-filsfils-
spring-segment-routing-policy-01 (work in progress), July
2017.

[I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d.,
daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R.,
Matsushima, S., Lebrun, D., Decraene, B., Peirens, B.,
Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P.,
Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W.,
Bashandy, A., Raza, K., Dukes, D., Clad, F., and P.
Camarillo, "SRv6 Network Programming", draft-filsfils-
spring-srv6-network-programming-01 (work in progress),
June 2017.

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

Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B.,
daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d.,
Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi,
T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk,
"IPv6 Segment Routing Header (SRH)", draft-ietf-6man-
segment-routing-header-07 (work in progress), July 2017.

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

Filsfils, C., Previdi, S., Bashandy, A., Decraene, B.,
Litkowski, S., and R. Shakir, "Segment Routing with MPLS
data plane", draft-ietf-spring-segment-routing-mpls-10
(work in progress), June 2017.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function
Chaining (SFC) Architecture", RFC 7665,
DOI 10.17487/RFC7665, October 2015,
<<https://www.rfc-editor.org/info/rfc7665>>.

Authors' Addresses

Francois Clad (editor)
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Chaitanya Yadlapalli
AT&T
USA

Email: cy098d@att.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Dawra
Cisco Systems, Inc.
USA

Email: gdawra@cisco.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: July 8, 2018

G. Dawra, Ed.
C. Filsfils
Cisco Systems
D. Bernier
Bell Canada
J. Uttaro
AT&T
B. Decraene
Orange
H. Elmalky
Ericsson
X. Xu
Huawei
F. Clad
K. Talaulikar
Cisco Systems
January 4, 2018

BGP Control Plane Extensions for Segment Routing based Service Chaining
draft-dawra-idr-bgp-sr-service-chaining-02

Abstract

The BGP Control Plane for the SR service-chaining solution is consistent with the BGP Control Plane for the topological Segment Routing Traffic Engineering (SR-TE) solution.

- o BGP Link-State(BGP-LS) address-family/sub-address-family[RFC7752] is used to discover service and topological characteristics from the network.
- o SR-TE policies[I-D.ietf-idr-segment-routing-te-policy] instantiate source-routed policies that may mix service and topological segments.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 8, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. BGP-LS Extensions for Service Chaining	3
3. Illustration	6
4. IANA Considerations	7
4.1. Service Type Table	7
4.2. Segment routing function Identifier(SFI)	8
5. Manageability Considerations	8
6. Operational Considerations	8
6.1. Operations	8
7. Security Considerations	8
8. Conclusions	8
9. Acknowledgements	9
10. References	9
10.1. Normative References	9
10.2. Informative References	10
Authors' Addresses	12

1. Introduction

Segments are introduced in the SR architecture [I-D.ietf-spring-segment-routing]. Segment Routing based Service chaining is well described in Section 6 of [I-D.clad-spring-segment-routing-service-chaining] document with an example network and services.

This document extend the example to add a Segment Routing Controller (SR-C) to the network, for the purpose of service discovery and SR policy instantiation.

Consider the network represented in Figure 1 below where:

- o A and B are two end hosts using IPv4.
- o S1 is an SR-aware firewall Service.
- o S2 is an SR-unaware DPI Service.

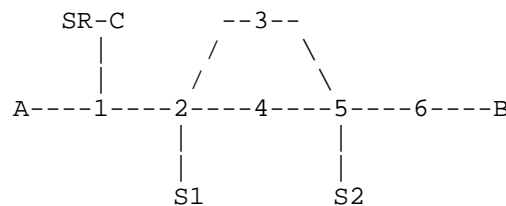


Figure 1: Network with Services

SR Controller (SR-C) is connected to Node 1, but may be attached to any node 1-6 in the network.

SR-C is capable of receiving BGP-LS updates to discover topology, and calculating constrained paths between 1 and 6.

However, if SR-C is configured to computation a constrained path from 1 and 6, including a DPI service (i.e., S2) it is not yet possible due to the lack of service distribution. SR-C does not know where a DPI Service is nor the SID for it. It does not know that S2 is a service it needs.

This document proposes an extension to BGP-LS for Service Chaining to distribute the service information to SR-C. There may be other alternate mechanisms to distribute service information to SR-C and are outside of scope of this document. There are no extensions required in SR-TE Policy SAFI.

2. BGP-LS Extensions for Service Chaining

For an attached service, following data needs to be shared with SR-C:

- o Service SID value (e.g. MPLS label or IPv6 address). Service SID MAY only be encoded as LOC:FUNCT, where LOC is the L most significant bits and FUNCT is the 128-L least significant

bits[I-D.filsfils-spring-srv6-network-programming]. ARGs bits, if any, MAY be set to 0 in the advertised service SID.

- o Function Identifier (Static Proxy, Dynamic Proxy, Shared Memory Proxy, Masquerading Proxy, SR Aware Service etc).
- o Service Type (DPI, Firewall, Classifier, LB etc).
- o Traffic Type (IPv4 OR IPv6 OR Ethernet)
- o Opaque Data (Such as brand and version, other extra information)

[I-D.clad-spring-segment-routing-service-chaining] defines SR-aware and SR-unaware services. This document will reuse these definitions. Per [RFC7752] Node Attributes are ONLY associated with the Node NLRI. All non-VPN information SHALL be encoded using AFI 16388 / SAFI 71. VPN information SHALL be encoded using AFI 16388 / SAFI 72 with associated RTs.

This document extends SRv6 Node SID TLV [I-D.dawra-idr-bgpls-srv6-ext] and SR-MPLS SID/Label TLV [I-D.ietf-idr-bgp-ls-segment-routing-ext] to associate the Service SID Value with Service-related Information using Service Chaining(SC) Sub-TLV.

Function Sub-TLV [I-D.dawra-idr-bgpls-srv6-ext] of Node SID TLV encodes Identifier(Function ID) along with associated Function Flags.

A Service Chaining (SC) Sub-TLV in Figure 2 is defined as:

	Type (2 octet)	

	Length (2 octet)	

	Service Type(ST) (2 octet)	

	Flags (1 octet)	

	Traffic Type(1 octet)	

	RESERVED (2 octet)	

Figure 2: Service Chaining(SC) Sub-TLV

Where:

Type: 16 bit field. TBD

Length: 16 bit field. The total length of the value portion of the TLV.

Service Type(ST): 16bit field. Service Type: categorizes the Service: (such as "Firewall", "Classifier" etc).

Flags: 8 bit field. Bits SHOULD be 0 on transmission and MUST be ignored on reception.

Traffic Type: 8 Bit field. A bit to identify if Service is IPv4 OR IPv6 OR L2 Ethernet Capable. Where:

Bit 0(LSB): Set to 1 if Service is IPv4 Capable

Bit 1: Set to 1 if Service is IPv6 Capable

Bit 2: Set to 1 if Service is Ethernet Capable

RESERVED: 16bit field. SHOULD be 0 on transmission and MUST be ignored on reception.

Service Type(ST) MUST be encoded as part of SC Sub-TLV.

There may be multiple instances of similar Services that needs to be distinguished. For example, firewalls made by different vendors A and B may need to be identified differently because, while they have similar functionality, their behavior is not identical.

In order for SDN Controller to identify the categories of Services and their associated SIDs, this section defines the BGP-LS extensions required to encode these characteristics and other relevant information about these Services.

Another Optional Opaque Metadata(OM) Sub-TLV of Node SID TLV may encode vendor specific information. Multiple of OM Sub-TLVs may be encoded.

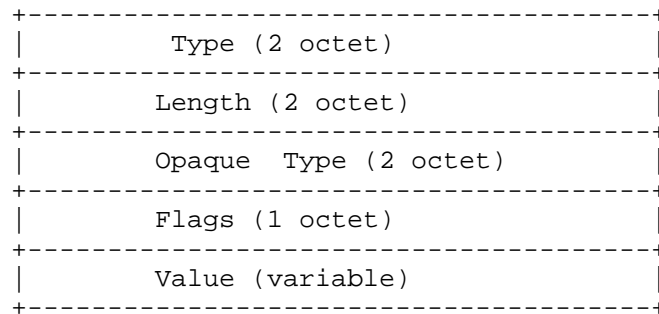


Figure 3: Opaque Metadata(OM) Sub-TLV

- o Type: 16 bit field. TBD.
- o Length: 16 bit field. The total length of the value portion of the TLV.
- o Opaque Type: 8-bit field. Only publishers and consumers of the opaque data are supposed to understand the data.
- o Flags: 8 bit field. Bits SHOULD be 0 on transmission and MUST be ignored on reception.
- o Value: Variable Length. Based on the data being encoded and length is recorded in length field.

Opaque Metadata(OM) Sub-TLV defined in Figure 3 may encode propriety or Service Opaque information such as:

- o Vendor specific Service Information.
- o Traffic Limiting Information to particular Service Type.
- o Opaque Information unique to the Service
- o Propriety Enterprise Service specific Information.

3. Illustration

In our SRv6 example above Figure 1 , Node 5 is configured with an SRv6 dynamic proxy segments (End.AD) C5::AD:F2 for S2.

The BGP-LS advertisement MUST contain and Node SID TLV:

- o Service SID: C5::AD:F2 SID

- o Function ID: END.AD

The BGP-LS advertisement MUST contain a SC Sub-TLV with:

- o Service Type: Deep Packet Inspection(DPI)
- o Traffic Type: IPv4 Capable.

The BGP-LS advertisement MAY contain a OM Sub-TLV with:

- o Opaque Type: Cisco DPI Version
- o Value: 3.5

In our example in Figure 1, using BGP SR-TE SAFI Update [I-D.ietf-idr-segment-routing-te-policy], SR Controller computes the candidate path and pushes the Policy.

SRv6 encapsulation policy < CF1::, C3::, C5::AD:F2, C6::D4:B > is signaled to Node 1 which has mix of service and topological segments.

4. IANA Considerations

This document requests assigning code-points from the registry "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs".

4.1. Service Type Table

IANA is request to create a new top-level registry called "Service Type Table (STT)". Valid values are in the range 0 to 65535. Values 0 and 65535 are to be marked "Reserved, not to be allocated".

Service Value(TBD)	Service	Reference	Date
32	Classifier	ref-to-set	date-to-set
33	Firewall	ref-to-set	date-to-set
34	Load Balancer	ref-to-set	date-to-set
35	DPI	ref-to-set	date-to-set

Figure 4

4.2. Segment routing function Identifier(SFI)

IANA is request to extend a top-level registry called "Segment Routing Function Identifier(SFI)" with new code points. This document extends the SFI values defined in [I-D.dawra-idr-bgpls-srv6-ext]. Details about the Service functions are defined in[I-D.clad-spring-segment-routing-service-chaining].

Function	Function Identifier
Static Proxy	8
Dynamic Proxy	9
Shared Memory Proxy	10
Masquerading Proxy	11
SRv6 Aware Service	12

5. Manageability Considerations

This section is structured as recommended in[RFC5706]

6. Operational Considerations

6.1. Operations

Existing BGP and BGP-LS operational procedures apply. No additional operation procedures are defined in this document.

7. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See the 'Security Considerations' section of [RFC4271]for a discussion of BGP security. Also refer to[RFC4272]and[RFC6952]for analysis of security issues for BGP.

8. Conclusions

This document proposes extensions to the BGP-LS to allow discovery of Services using Segment Routing.

9. Acknowledgements

The authors would like to thank Krishnaswamy Ananthamurthy for his review of this document.

10. References

10.1. Normative References

- [I-D.clad-spring-segment-routing-service-chaining]
Clad, F., Filsfils, C., Camarillo, P.,
daniel.bernier@bell.ca, d., Decraene, B., Peirens, B.,
Yadlapalli, C., Xu, X., Salsano, S., Abdelsalam, A., and
G. Dawra, "Segment Routing for Service Chaining", draft-
clad-spring-segment-routing-service-chaining-00 (work in
progress), October 2017.
- [I-D.dawra-idr-bgpls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Sreekantiah, A.,
and L. Ginsberg, "BGP Link State extensions for IPv6
Segment Routing(SRv6)", draft-dawra-idr-bgpls-srv6-ext-00
(work in progress), October 2017.
- [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>>.
- [RFC5706] Harrington, D., "Guidelines for Considering Operations and
Management of New Protocols and Protocol Extensions",
RFC 5706, DOI 10.17487/RFC5706, November 2009,
<<https://www.rfc-editor.org/info/rfc5706>>.
- [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>>.
- [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>>.

10.2. Informative References

- [I-D.dawra-bgp-srv6-vpn]
(Unknown), (., Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., and S. Matsushima, "BGP Signaling of IPv6-Segment-Routing-based VPN Networks", draft-dawra-bgp-srv6-vpn-00 (work in progress), March 2017.
- [I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Raza, K., Liste, J., Clad, F., Talaulikar, K., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., and P. Mattes, "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy-04 (work in progress), December 2017.
- [I-D.filsfils-spring-srv6-network-programming]
Filsfils, C., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Matsushima, S., Lebrun, D., Decraene, B., Peirens, B., Salsano, S., Naik, G., Elmalky, H., Jonnalagadda, P., Sharif, M., Ayyangar, A., Mynam, S., Henderickx, W., Bashandy, A., Raza, K., Dukes, D., Clad, F., and P. Camarillo, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-03 (work in progress), December 2017.
- [I-D.ietf-6man-segment-routing-header]
Previdi, S., Filsfils, C., Raza, K., Leddy, J., Field, B., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Matsushima, S., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., Lebrun, D., Steinberg, D., and R. Raszuk, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-07 (work in progress), July 2017.
- [I-D.ietf-bess-evpn-prefix-advertisement]
Rabadan, J., Henderickx, W., Drake, J., Lin, W., and A. Sajassi, "IP Prefix Advertisement in EVPN", draft-ietf-bess-evpn-prefix-advertisement-09 (work in progress), November 2017.

- [I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Psenak, P., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-03 (work in progress), July 2017.
- [I-D.ietf-idr-bgp-prefix-sid]
Previdi, S., Filsfils, C., Lindem, A., Sreekantiah, A., and H. Gredler, "Segment Routing Prefix SID extensions for BGP", draft-ietf-idr-bgp-prefix-sid-08 (work in progress), January 2018.
- [I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-01 (work in progress), December 2017.
- [I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-15 (work in progress), December 2017.
- [I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-14 (work in progress), December 2017.
- [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>>.
- [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>>.

[RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<https://www.rfc-editor.org/info/rfc5549>>.

Authors' Addresses

Gaurav Dawra (editor)
Cisco Systems
USA

Email: gdawra.ietf@gmail.com

Clarence Filsfils
Cisco Systems
Belgium

Email: cfilsfil@cisco.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Jim Uttaro
AT&T
USA

Email: jul738@att.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Hani Elmalky
Ericsson
USA

Email: hani.elmalky@gmail.com

Xiaohu Xu
Huawei

Email: xuxiaohu@huawei.com

Francois Clad
Cisco Systems
France

Email: fclad@cisco.com

Ketan Talaulikar
Cisco Systems
India

Email: ketant@cisco.com

Inter-Domain Routing
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2019

G. Dawra, Ed.
LinkedIn
C. Filsfils
K. Talaulikar, Ed.
Cisco Systems
M. Chen
Huawei
D. Bernier
Bell Canada
J. Uttaro
AT&T
B. Decraene
Orange
H. Elmalky
Ericsson
March 24, 2019

BGP Link State Extensions for SRv6
draft-dawra-idr-bgpls-srv6-ext-06

Abstract

Segment Routing IPv6 (SRv6) allows for a flexible definition of end-to-end paths within various topologies by encoding paths as sequences of topological or functional sub-paths, called "segments". These segments are advertised by the various protocols such as BGP, ISIS and OSPFv3.

BGP Link-state (BGP-LS) address-family solution for SRv6 is similar to BGP-LS for SR for MPLS dataplane. This draft defines extensions to the BGP-LS to advertise SRv6 Segments along with their functions and other attributes via BGP.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on September 25, 2019.

Copyright Notice

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

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

Table of Contents

1. Introduction	3
2. BGP-LS Extensions for SRv6	4
3. SRv6 Node Attributes	5
3.1. SRv6 Capabilities TLV	5
3.2. SRv6 Node MSD Types	6
4. SRv6 Link Attributes	7
4.1. SRv6 End.X SID TLV	7
4.2. SRv6 LAN End.X SID TLV	9
4.3. SRv6 Link MSD Types	11
5. SRv6 Prefix Attributes	12
5.1. SRv6 Locator TLV	12
6. SRv6 SID NLRI	14
6.1. SRv6 SID Information TLV	15
7. SRv6 SID Attributes	16
7.1. SRv6 Endpoint Function TLV	16
7.2. SRv6 BGP Peer Node SID TLV	17
8. IANA Considerations	19
8.1. BGP-LS NLRI-Types	19
8.2. BGP-LS TLVs	19

9. Manageability Considerations	20
10. Operational Considerations	20
10.1. Operations	20
11. Security Considerations	20
12. Contributors	20
13. Acknowledgements	21
14. References	21
14.1. Normative References	21
14.2. Informative References	23
Authors' Addresses	23

1. Introduction

SRv6 refers to Segment Routing instantiated on the IPv6 dataplane [RFC8402]. Segment Identifier (SID) is often used as a shorter reference for "SRv6 Segment".

The network programming paradigm [I-D.filsfils-spring-srv6-network-programming] is central to SRv6. It describes how different functions can be bound to their SIDs and how a network program can be expressed as a combination of SIDs.

An SRv6-capable node N maintains a "My SID Table" (refer [I-D.filsfils-spring-srv6-network-programming]). This table contains all the SRv6 segments explicitly instantiated at node N.

The IS-IS [I-D.bashandy-isis-srv6-extensions] and OSPFv3 [I-D.li-ospf-ospfv3-srv6-extensions] link-state routing protocols have been extended to advertise some of these SRv6 SIDs and SRv6-related information. BGP ([I-D.dawra-idr-srv6-vpn]) has been extended to advertise some of these SRv6 SIDs for VPN services. Certain other SRv6 SIDs may be instantiated on a node via other mechanisms for topological or service functionalities.

The advertisement of SR related information along with the topology for the MPLS dataplane instantiation is specified in [I-D.ietf-idr-bgp-ls-segment-routing-ext] and for the BGP Egress Peer Engineering (EPE) is specified in [I-D.ietf-idr-bgpls-segment-routing-epe]. On the similar lines, introducing the SRv6 related information in BGP-LS allows it's consumer applications that require topological visibility to also receive the "My SID Table" from nodes across a domain or even across Autonomous Systems (AS), as required. This allows applications to leverage the SRv6 capabilities for network programming.

The identifying key of each Link-State object, namely a node, link, or prefix, is encoded in the NLRI and the properties of the object are encoded in the BGP-LS Attribute [RFC7752].

This document describes extensions to BGP-LS to advertise the SRv6 "My SID Table" and other SRv6 information from all the SRv6 capable nodes in the domain when sourced from link-state routing protocols and directly from individual SRv6 capable nodes when sourced from BGP.

2. BGP-LS Extensions for SRv6

BGP-LS[RFC7752] defines the BGP Node, Link and Prefix attributes. All non-VPN link, node, and prefix information SHALL be encoded using AFI 16388 / SAFI 71. VPN link, node, and prefix information SHALL be encoded using AFI 16388 / SAFI 72.

The SRv6 information pertaining to a node is advertised via the BGP-LS Node NLRI and using the BGP-LS Attribute TLVs as follows:

- o SRv6 Capabilities of the node is advertised via a new SRv6 Capabilities TLV
- o New MSD types introduced for SRv6 are advertised as new sub-TLVs of the Node MSD TLV specified in [I-D.ietf-idr-bgp-ls-segment-routing-msd].
- o Algorithm support for SRv6 is advertised via the existing SR Algorithm TLV specified in [I-D.ietf-idr-bgp-ls-segment-routing-ext].

The SRv6 information pertaining to a link is advertised via the BGP-LS Link NLRI and using the BGP-LS Attribute TLVs as follows:

- o SRv6 End.X SID of the link state routing adjacency or the BGP EPE Peer Adjacency is advertised via a new SRv6 End.X SID TLV
- o SRv6 LAN End.X SID of the link state routing adjacency to a non-DR/DIS router is advertised via a new SRv6 LAN End.X SID TLV
- o New MSD types introduced for SRv6 are advertised as new sub-TLVs of the Link MSD TLV specified in [I-D.ietf-idr-bgp-ls-segment-routing-msd].

The SRv6 Locator information of a node is advertised via the BGP-LS Prefix NLRI using the new SRv6 Locator TLV in the BGP-LS Attribute.

The SRv6 SIDs associated with the node from its "My SID Table" are advertised as a newly introduce BGP-LS SRv6 SID NLRI. This enables the BGP-LS encoding to scale to cover a potentially large set of SRv6 SIDs instantiated on a node with the granularity of individual SIDs and without affecting the size and scalability of the BGP-LS updates.

New BGP-LS Attribute TLVs are introduced for the SRv6 SID NLRI as follows:

- o The endpoint function of the SRv6 SID is advertised via a new SRv6 Endpoint Function TLV
- o The BGP EPE Peer Node and Peer Set SID context is advertised via a new SRv6 BGP EPE Peer Node SID TLV

When the BGP-LS router is advertising topology information that it sources from the underlying link-state routing protocol, then it maps the corresponding SRv6 information from the SRv6 extensions for IS-IS [I-D.bashandy-isis-srv6-extensions] and OSPFv3 [I-D.li-ospf-ospfv3-srv6-extensions] protocols to their BGP-LS TLVs/sub-TLVs for all SRv6 capable nodes in that routing protocol domain. When the BGP-LS router is advertising topology information from the BGP routing protocol [I-D.ietf-idr-bgpls-segment-routing-epe], then it advertises the SRv6 information from the local node alone (e.g. BGP EPE topology information or in the case of a data center network running BGP as the only routing protocol).

Subsequent sections of this document specify the encoding of the newly defined extensions.

3. SRv6 Node Attributes

SRv6 attributes of a node are advertised using the new BGP-LS Attribute TLVs defined in this section and associated with the BGP-LS Node NLRI.

3.1. SRv6 Capabilities TLV

This BGP-LS Attribute TLV is used to announce the SRv6 capabilities of the node along with the BGP-LS Node NLRI and indicates the SRv6 support by the node. A single instance of this TLV MUST be included in the BGP-LS attribute for each SRv6 capable node. This TLV maps to the SRv6 Capabilities sub-TLV and the SRv6 Capabilities TLV of the IS-IS and OSPFv3 protocol SRv6 extensions respectively.

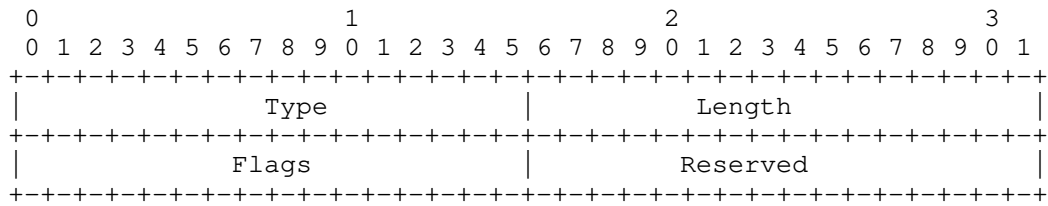


Figure 1: SRv6 Capabilities TLV Format

Where:

- o Type: 2 octet field with value TBD, see Section 8.
- o Length : 2 octet field with value set to 4.
- o Flags: 2 octet field. The following flags are defined:

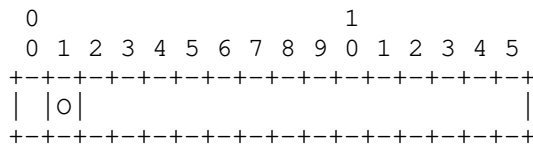


Figure 2: SRv6 Capability TLV Flags Format

- * O-flag: If set, then router is capable of supporting SRH O-bit Flags, as specified in [I-D.ali-spring-srv6-oam].
- o Reserved: 2 octet that SHOULD be set to 0 and MUST be ignored on receipt.

3.2. SRv6 Node MSD Types

The Node MSD TLV [I-D.ietf-idr-bgp-ls-segment-routing-msd] of the BGP-LS Attribute of the Node NLRI is also used to advertise the limits and the supported Segment Routing Header (SRH) [I-D.ietf-6man-segment-routing-header] operations supported by the SRv6 capable node. The SRv6 MSD Types specified in [I-D.bashandy-isis-srv6-extensions] are also used with the BGP-LS Node MSD TLV as these codepoints are shared between IS-IS, OSPF and BGP-LS protocols. The description and semantics of these new MSD types for BGP-LS are identical as specified [I-D.bashandy-isis-srv6-extensions] and summarized in the table below:

MSD Type	Description
TBD	Maximum Segments Left
TBD	Maximum End Pop
TBD	Maximum T.Insert
TBD	Maximum T.Encaps
TBD	Maximum End D

Figure 3: SRv6 Node MSD Types

Each MSD type is encoded as a one octet type followed by a one octet value.

4. SRv6 Link Attributes

SRv6 attributes and SIDs associated with a link or adjacency are advertised using the new BGP-LS Attribute TLVs defined in this section and associated with the BGP-LS Link NLRI.

4.1. SRv6 End.X SID TLV

The SRv6 End.X SID TLV is used to advertise the SRv6 End.X SIDs that correspond to a point-to-point or point-to-multipoint link or adjacency of the local node for IS-IS and OSPFv3 protocols. This TLV can also be used to advertise the End.X function SRv6 SID corresponding to the underlying layer-2 member links for a layer-3 bundle interface using L2 Bundle Member Attribute TLV as specified in [I-D.ietf-idr-bgp-ls-segment-routing-ext] .

For the nodes running BGP routing protocol, this TLV is used to advertise the BGP EPE Peer Adjacency SID for SRv6 on the same lines as specified for SR/MPLS in [I-D.ietf-idr-bgpls-segment-routing-epe]. The SRv6 End.X SID for the BGP Peer Adjacency indicates the cross-connect to a specific layer-3 link to the specific BGP session peer (neighbor).

The TLV has the following format:

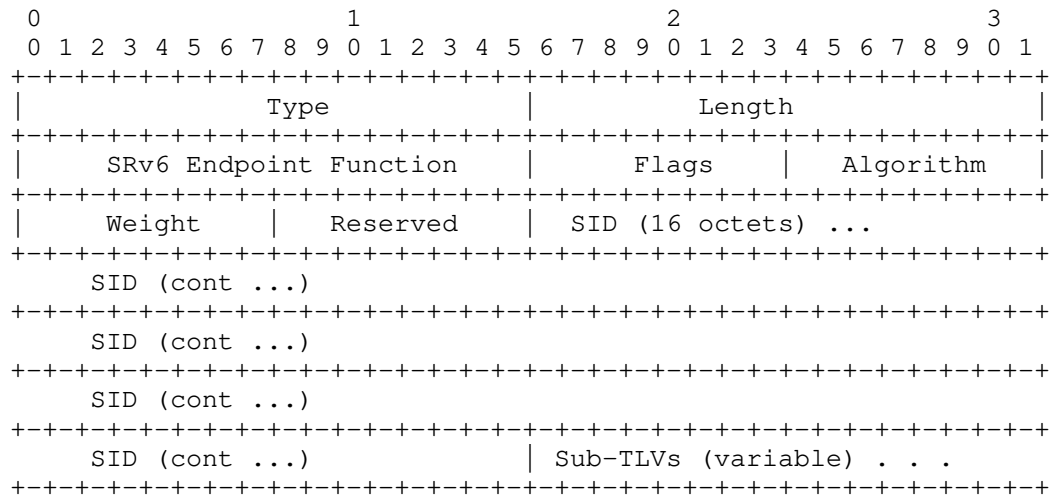


Figure 4: SRv6 End.X TLV Format

Where:

Type: 2 octet field with value TBD, see Section 8.

Length: 2 octet field with the total length of the value portion of the TLV.

Function Code: 2 octet field. The Endpoint Function code point for this SRv6 SID as defined in [I-D.filsfils-spring-srv6-network-programming].

Flags: 1 octet of flags with the following definition:

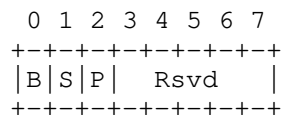


Figure 5: SRv6 End.X SID TLV Flags Format

- * B-Flag: Backup Flag. If set, the SID is eligible for protection (e.g. using IPFRR) as described in [RFC8355].
- * S-Flag: Set Flag. When set, the S-Flag indicates that the SID refers to a set of adjacencies (and therefore MAY be assigned to other adjacencies as well).

- * P-Flag: Persistent Flag: When set, the P-Flag indicates that the SID is persistently allocated, i.e., the value remains consistent across router restart and/or interface flap.
- * Rsvd bits: Reserved for future use and MUST be zero when originated and ignored when received.

Algorithm: 1 octet field. Algorithm associated with the SID. Algorithm values are defined in the IGP Algorithm Type registry.

Weight: 1 octet field. The value represents the weight of the SID for the purpose of load balancing. The use of the weight is defined in [RFC8402].

Reserved: 1 octet field that SHOULD be set to 0 and MUST be ignored on receipt.

SID: 16 octet field. This field encodes the advertised SRv6 SID as 128 bit value.

Sub-TLVs : currently none defined. Used to advertise sub-TLVs that provide additional attributes for the given SRv6 End.X SID.

4.2. SRv6 LAN End.X SID TLV

For a LAN interface, normally a node only announces its adjacency to the IS-IS pseudo-node (or the equivalent OSPF Designated Router). The SRv6 LAN End.X SID TLV allows a node to announce SRv6 SID corresponding to functions like END.X for its adjacencies to all other (i.e. non-DIS or non-DR) nodes attached to the LAN in a single instance of the BGP-LS Link NLRI. Without this TLV, the corresponding BGP-LS link NLRI would need to be originated for each additional adjacency in order to advertise the SRv6 End.X SID TLVs for these neighbor adjacencies.

The IS-IS and OSPFv3 SRv6 LAN End.X SID TLVs have the following format:

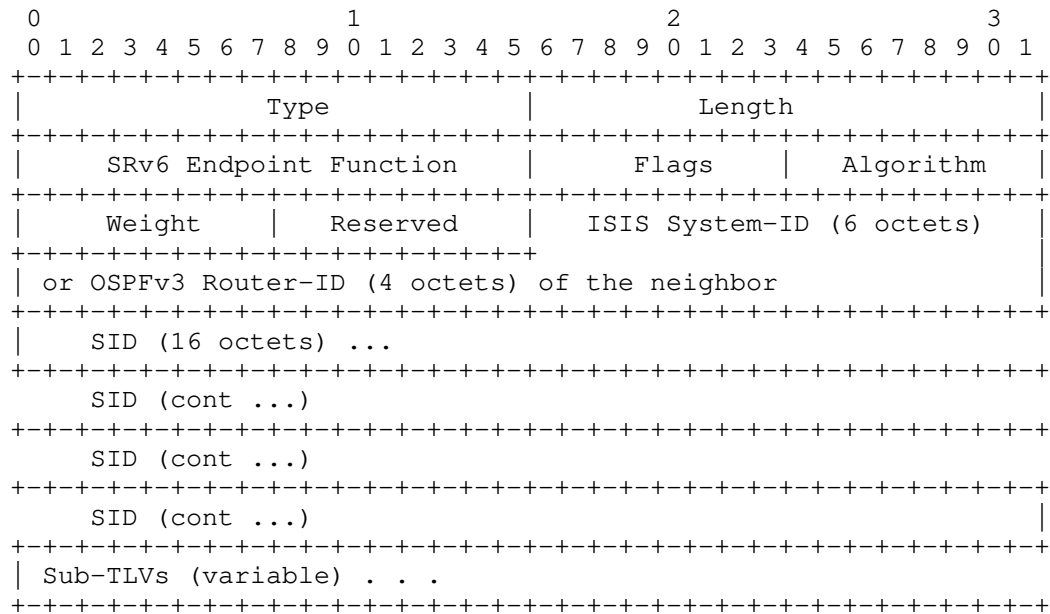


Figure 6: SRv6 LAN End.X SID TLV Format

Where:

- o Type: 2 octet field with value TBD in case of IS-IS and TBD in case of OSPFv3, see Section 8.
- o Length: 2 octet field with the total length of the value portion of the TLV.
- o Function Code: 2 octet field. The Endpoint Function code point for this SRv6 SID as defined in [I-D.filsfils-spring-srv6-network-programming].
- o Flags: 1 octet of flags with the following definition:

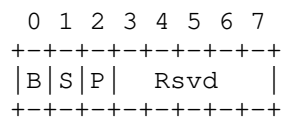


Figure 7: SRv6 LAN End.X SID TLV Flags Format

- * B-Flag: Backup Flag. If set, the SID is eligible for protection (e.g. using IPFRR) as described in [RFC8355].
 - * S-Flag: Set Flag. When set, the S-Flag indicates that the SID refers to a set of adjacencies (and therefore MAY be assigned to other adjacencies as well).
 - * P-Flag: Persistent Flag: When set, the P-Flag indicates that the SID is persistently allocated, i.e., the value remains consistent across router restart and/or interface flap.
 - * Rsvd bits: Reserved for future use and MUST be zero when originated and ignored when received.
- o Algorithm: 1 octet field. Algorithm associated with the SID. Algorithm values are defined in the IGP Algorithm Type registry.
 - o Weight: 1 octet field. The value represents the weight of the SID for the purpose of load balancing. The use of the weight is defined in [RFC8402].
 - o Reserved: 1 octet field that SHOULD be set to 0 and MUST be ignored on receipt.
 - o Neighbor ID : 6 octets of ISIS System ID of the neighbor for the ISIS SRv6 LAN End.X SID TLV and 4 octets of OSPFv3 Router-id of the neighbor for the OSPFv3 SRv6 LAN End.X SID TLV.
 - o SID: 16 octet field. This field encodes the advertised SRv6 SID as 128 bit value.
 - o Sub-TLVs : currently none defined. Used to advertise sub-TLVs that provide additional attributes for the given SRv6 LAN End.X SID.

4.3. SRv6 Link MSD Types

The Link MSD TLV [I-D.ietf-idr-bgp-ls-segment-routing-msd] of the BGP-LS Attribute of the Link NLRI is also used to advertise the limits and the supported Segment Routing Header (SRH) operations supported on the specific link by the SRv6 capable node. The SRv6 MSD Types specified in [I-D.bashandy-isis-srv6-extensions] are also used with the BGP-LS Link MSD TLV as these codepoints are shared between IS-IS, OSPF and BGP-LS protocols. The description and semantics of these new MSD types for BGP-LS are identical as specified [I-D.bashandy-isis-srv6-extensions] and summarized in the table below:

MSD Type	Description
TBD	Maximum Segments Left
TBD	Maximum End Pop
TBD	Maximum T.Insert
TBD	Maximum T.Encaps
TBD	Maximum End D

Figure 8: SRv6 Link MSD Types

Each MSD type is encoded as a one octet type followed by a one octet value.

5. SRv6 Prefix Attributes

SRv6 attributes with an IPv6 prefix are advertised using the new BGP-LS Attribute TLVs defined in this section and associated with the BGP-LS Prefix NLRI.

5.1. SRv6 Locator TLV

As described in [I-D.filsfils-spring-srv6-network-programming], an SRv6 SID is 128 bits and represented as

LOC:FUNCT

where LOC (the locator portion) is the L most significant bits and FUNCT is the 128-L least significant bits. L is called the locator length and is flexible. A node is provisioned with one or more locators supported by that node. Locators are covering prefixes for the set of SIDs provisioned on that node. These Locators are advertised as BGP-LS Prefix NLRI objects along with the SRv6 Locator TLV in its BGP-LS Attribute.

The IPv6 Prefix matching the Locator MAY be also advertised as a prefix reachability by the underlying routing protocol. In this case, the Prefix NLRI would be also associated with the Prefix Metric TLV that carries the routing metric for this prefix. When the Locator prefix is not being advertised as a prefix reachability, then the Prefix NLRI would have the SRv6 Locator TLV associated with it but no Prefix Metric TLV. In the absence of Prefix Metric TLV, the consumer of the BGP-LS topology information MUST NOT interpret the Locator prefix as a prefix reachability routing advertisement.

The SRv6 Locator TLV has the following format:

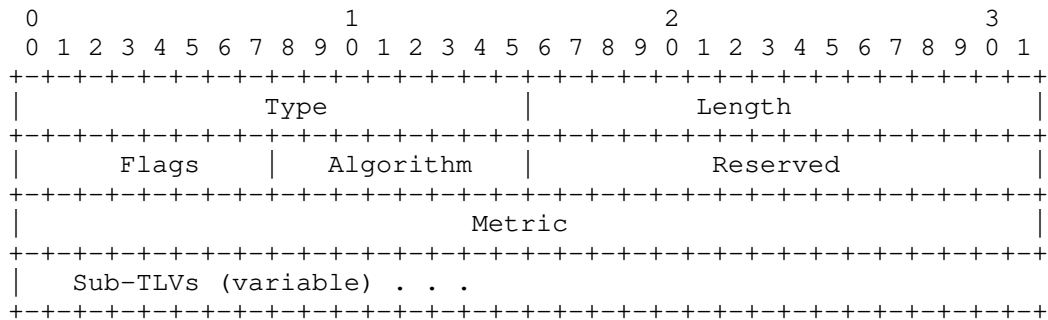


Figure 9: SRv6 Locator TLV Format

Where:

Type: 2 octet field with value TBD, see Section 8.

Length: 2 octet field with the total length of the value portion of the TLV.

Flags: 1 octet of flags with the following definition:

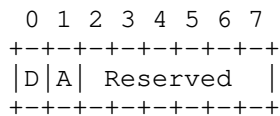


Figure 10: SRv6 Locator TLV Flags Format

- * D-Flag: Indicates that the locator has been leaked into the IGP domain when set. IS-IS operations for this are discussed in [I-D.bashandy-isis-srv6-extensions].
- * A-Flag: When the Locator is associated with anycast destinations, the A flag SHOULD be set. Otherwise, this bit MUST be clear.
- * Reserved bits: Reserved for future use and MUST be zero when originated and ignored when received.

Algorithm: 1 octet field. Algorithm associated with the SID. Algorithm values are defined in the IGP Algorithm Type registry.

Reserved: 2 octet field. The value MUST be zero when originated and ignored when received.

Metric: 4 octet field. The value of the metric for the Locator.

Sub-TLVs : currently none defined. Used to advertise sub-TLVs that provide additional attributes for the given SRv6 Locator.

6. SRv6 SID NLRI

SRv6 SID information is advertised in BGP UPDATE messages using the MP_REACH_NLRI and MP_UNREACH_NLRI attributes [RFC4760]. The "Link-State NLRI" defined in [RFC7752] is extended to carry the SRv6 SID information.

A new "Link-State NLRI Type" is defined for SRv6 SID information as following:

- o Link-State NLRI Type: SRv6 SID NLRI (value TBD see IANA Considerations Section 8.1).

The format of this new NLRI type is as shown in the following figure:

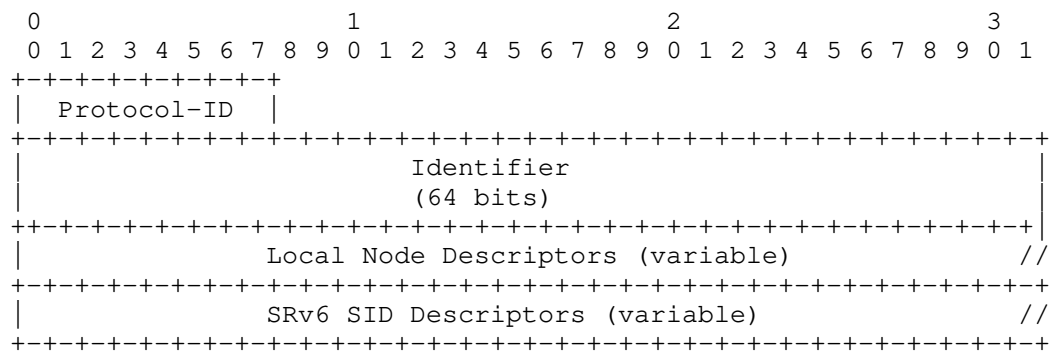


Figure 11: SRv6 SID NLRI Format

Where:

- o Protocol-ID: 1 octet field that specifies the protocol component through which BGP-LS learns the SRv6 SIDs of the node. The following Protocol-IDs apply to the SRv6 SID NLRI:

Protocol-ID	NLRI information source protocol
1	IS-IS Level 1
2	IS-IS Level 2
4	Direct
5	Static configuration
6	OSPFv3
7	BGP

Figure 12: Protocol IDs for SRv6 SID NLRI

- o Identifier: 8 octet value as defined in [RFC7752].
- o Local Node Descriptors TLV: as defined in [RFC7752] for IGPs, local and static configuration and as defined in [I-D.ietf-idr-bgpls-segment-routing-epe] for BGP protocol.
- o SRv6 SID Descriptors: MUST include the SRv6 SID Information TLV defined in Section 6.1 and optionally MAY include the Multi-Topology Identifier TLV as defined in [RFC7752].

New TLVs carried in the BGP Link State Attribute defined in [RFC7752] are also defined in order to carry the attributes of a SRv6 SID in Section 7.

6.1. SRv6 SID Information TLV

A SRv6 SID is a 128 bit value [I-D.filsfils-spring-srv6-network-programming] and is encoded using the SRv6 SID Information TLV.

The TLV has the following format:

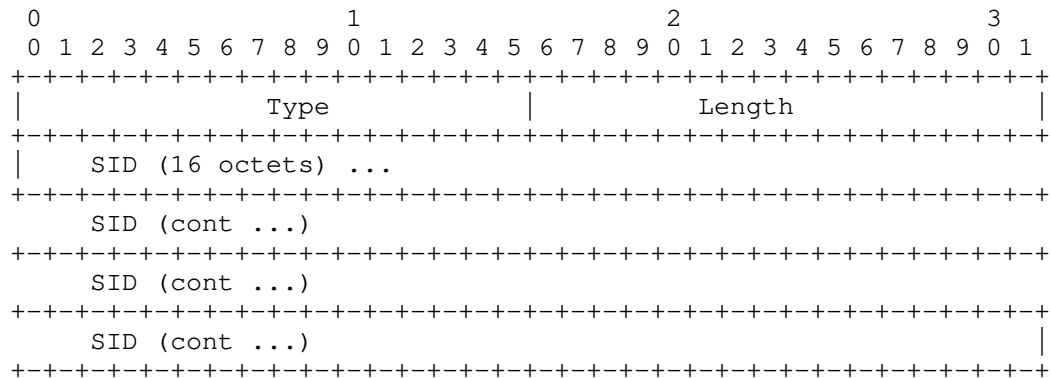


Figure 13: SRv6 SID Information TLV Format

Where:

Type: 2 octet field with value TBD, see Section 8.

Length: 2 octet field with value set to 16.

SID: 16 octet field. This field encodes the advertised SRv6 SID as 128 bit value.

7. SRv6 SID Attributes

This section specifies the new TLVs to be carried in the BGP Link State Attribute associated with the BGP-LS SRv6 SID NLRI.

7.1. SRv6 Endpoint Function TLV

Each SRv6 SID instantiated in the "My SID Table" of an SRv6 capable node has a specific instruction bound to it. A set of well-known functions that can be associated with a SID are defined in [I-D.filsfils-spring-srv6-network-programming].

The SRv6 Endpoint Function TLV is a mandatory TLV that MUST be included in the BGP-LS Attribute associated with the BGP-LS SRv6 SID NLRI. The TLV has the following format:

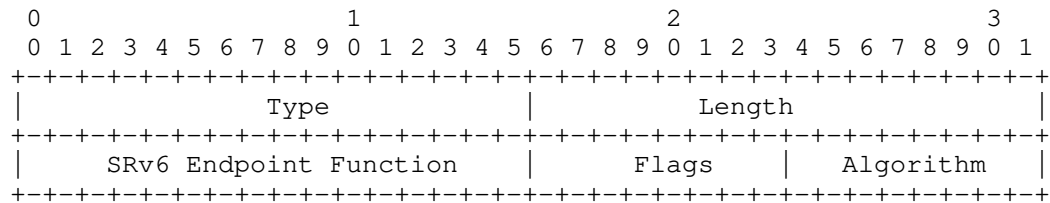


Figure 14: SRv6 Endpoint Function TLV

Where:

Type: 2 octet field with value TBD, see Section 8.

Length: 2 octet field with the value 4.

Function Code: 2 octet field. The Endpoint Function code point for this SRv6 SID as defined in [I-D.filsfils-spring-srv6-network-programming].

Flags: 1 octet of flags with the none defined currently. Reserved for future use and MUST be zero when originated and ignored when received.

Algorithm: 1 octet field. Algorithm associated with the SID. Algorithm values are defined in the IGP Algorithm Type registry.

7.2. SRv6 BGP Peer Node SID TLV

The BGP Peer Node SID and Peer Set SID for SR with MPLS dataplane are specified in [I-D.ietf-idr-bgpls-segment-routing-epe]. The similar Peer Node and Peer Set SID functionality can be realized with SRv6 using the END.X SRv6 SID. The SRv6 BGP Peer Node SID TLV is an optional TLV for use in the BGP-LS Attribute for an SRv6 SID NLRI corresponding to BGP protocol. This TLV MUST be included along with SRv6 End.X SID that is associated with the BGP Peer Node or Peer Set functionality.

The TLV has the following format:

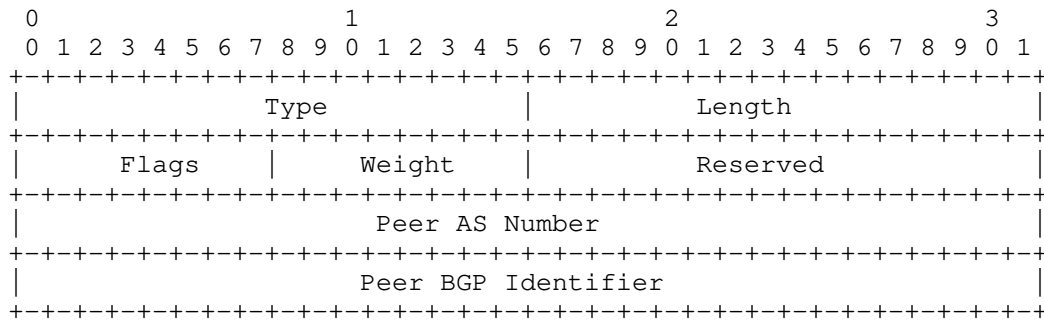


Figure 15: SRv6 BGP Peer Node SID TLV Format

Where:

- o Type: 2 octet field with value TBD, see Section 8.
- o Length: 2 octet field with the value 12.
- o Flags: 1 octet of flags with the following definition:

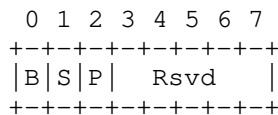


Figure 16: SRv6 BGP Peer End.X SID TLV Flags Format

- * B-Flag: Backup Flag. If set, the SID is eligible for protection (e.g. using IPFRR) as described in [RFC8355].
- * S-Flag: Set Flag. When set, the S-Flag indicates that the SID refers to a set of BGP peering sessions (i.e. BGP Peer Set SID functionality) and therefore MAY be assigned to one or more End.X SIDs associated with BGP peer sessions.
- * P-Flag: Persistent Flag: When set, the P-Flag indicates that the SID is persistently allocated, i.e., the value remains consistent across router restart and/or session flap.
- * Rsvd bits: Reserved for future use and MUST be zero when originated and ignored when received.

- o Weight: 1 octet field. The value represents the weight of the SID for the purpose of load balancing. The use of the weight is defined in [RFC8402].
- o Peer AS Number : 4 octets of BGP AS number of the peer router.
- o Peer BGP Identifier : 4 octets of the BGP Identifier (BGP Router-ID) of the peer router.

For a SRv6 BGP EPE Peer Node SID, one instance of this TLV is associated with the SRv6 SID. For SRv6 BGP EPE Peer Set SID, multiple instances of this TLV (one for each peer in the "peer set") are associated with the SRv6 SID and the S (set/group) flag is SET.

8. IANA Considerations

This document requests assigning code-points from the IANA "Border Gateway Protocol - Link State (BGP-LS) Parameters" registry as described in the sub-sections below.

8.1. BGP-LS NLRI-Types

The following codepoints is suggested (to be assigned by IANA) from within the sub-registry called "BGP-LS NLRI-Types":

Type	NLRI Type	Reference
6	SRv6 SID	this document

Figure 17: SRv6 SID NLRI Type Codepoint

8.2. BGP-LS TLVs

The following TLV codepoints are suggested (to be assigned by IANA) from within the sub-registry called "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs":

TLV Code Point	Description	Value defined in
TBD	SRv6 Capabilities TLV	this document
TBD	SRv6 End.X SID TLV	this document
TBD	IS-IS SRv6 LAN End.X SID TLV	this document
TBD	OSPFv3 SRv6 LAN End.X SID TLV	this document
TBD	SRv6 Locator TLV	this document
TBD	SRv6 SID Information TLV	this document
TBD	SRv6 Endpoint Function TLV	this document
TBD	SRv6 BGP Peer Node SID TLV	this document

Figure 18: SRv6 BGP-LS Attribute TLV Codepoints

9. Manageability Considerations

This section is structured as recommended in[RFC5706]

10. Operational Considerations

10.1. Operations

Existing BGP and BGP-LS operational procedures apply. No additional operation procedures are defined in this document.

11. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See the 'Security Considerations' section of [RFC4271] for a discussion of BGP security. Also refer to[RFC4272] and [RFC6952] for analysis of security issues for BGP.

12. Contributors

Arjun Sreekantiah
Individual
US

Ies Ginsberg
Cisco Systems
US
Email: ginsberg@cisco.com

Shunwan Zhuang
Huawei
China
Email: zhuangshunwan@huawei.com

13. Acknowledgements

The authors would like to thank Peter Psenak and Arun Babu for their review of this document and their comments.

14. References

14.1. Normative References

[I-D.ali-spring-srv6-oam]

Ali, Z., Filsfils, C., Kumar, N., Pignataro, C., faiqbal@cisco.com, f., Gandhi, R., Leddy, J., Matsushima, S., Raszuk, R., daniel.voyer@bell.ca, d., Dawra, G., Peirens, B., Chen, M., and G. Naik, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ali-spring-srv6-oam-02 (work in progress), October 2018.

[I-D.bashandy-isis-srv6-extensions]

Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-05 (work in progress), March 2019.

[I-D.dawra-idr-srv6-vpn]

Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., Matsushima, S., and S. Zhuang, "BGP Signaling for SRv6 based Services.", draft-dawra-idr-srv6-vpn-05 (work in progress), October 2018.

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

Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-filsfils-spring-srv6-network-programming-07 (work in progress), February 2019.

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

Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-16 (work in progress), February 2019.

- [I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H.,
and M. Chen, "BGP Link-State extensions for Segment
Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-12
(work in progress), March 2019.
- [I-D.ietf-idr-bgp-ls-segment-routing-msd]
Tantsura, J., Chunduri, U., Mirsky, G., Sivabalan, S., and
N. Triantafyllis, "Signaling MSD (Maximum SID Depth) using
Border Gateway Protocol Link-State", draft-ietf-idr-bgp-
ls-segment-routing-msd-04 (work in progress), February
2019.
- [I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray,
S., and J. Dong, "BGP-LS extensions for Segment Routing
BGP Egress Peer Engineering", draft-ietf-idr-bgpls-
segment-routing-epe-17 (work in progress), October 2018.
- [I-D.li-ospf-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak,
"OSPFv3 Extensions for SRv6", draft-li-ospf-
ospfv3-srv6-extensions-03 (work in progress), March 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and
S. Ray, "North-Bound Distribution of Link-State and
Traffic Engineering (TE) Information Using BGP", RFC 7752,
DOI 10.17487/RFC7752, March 2016,
<<https://www.rfc-editor.org/info/rfc7752>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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>>.

14.2. Informative References

- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis", RFC 4272, DOI 10.17487/RFC4272, January 2006, <<https://www.rfc-editor.org/info/rfc4272>>.
- [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>>.
- [RFC5706] Harrington, D., "Guidelines for Considering Operations and Management of New Protocols and Protocol Extensions", RFC 5706, DOI 10.17487/RFC5706, November 2009, <<https://www.rfc-editor.org/info/rfc5706>>.
- [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>>.
- [RFC8355] Filsfils, C., Ed., Previdi, S., Ed., Decraene, B., and R. Shakir, "Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks", RFC 8355, DOI 10.17487/RFC8355, March 2018, <<https://www.rfc-editor.org/info/rfc8355>>.

Authors' Addresses

Gaurav Dawra (editor)
LinkedIn
USA

Email: gdawra.ietf@gmail.com

Clarence Filsfils
Cisco Systems
Belgium

Email: cfilsfil@cisco.com

Ketan Talaulikar (editor)
Cisco Systems
India

Email: ketant@cisco.com

Mach Chen
Huawei
China

Email: mach.chen@huawei.com

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Jim Uttaro
AT&T
USA

Email: jul738@att.com

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Hani Elmalky
Ericsson
USA

Email: hani.elmalky@gmail.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 4, 2018

J. Dong
M. Chen
D. Dhody
Huawei Technologies
J. Tantsura
Individual
K. Kumaki
KDDI Corporation
T. Murai
Furukawa Network Solution Corp.
July 3, 2017

BGP Extensions for Path Computation Element (PCE) Discovery
draft-dong-pce-discovery-proto-bgp-07

Abstract

In networks where a Path Computation Element (PCE) is used for path computation, it is desirable for the Path Computation Clients (PCCs) to discover dynamically and automatically a set of PCEs along with certain information relevant for PCE selection. RFC 5088 and RFC 5089 define the PCE discovery mechanisms based on Interior Gateway Protocols (IGP). This document defines extensions to BGP for the advertisement of PCE Discovery information. The BGP based PCE discovery mechanism is complementary to the existing IGP based mechanisms.

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 <http://datatracker.ietf.org/drafts/current/>.

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

This Internet-Draft will expire on January 4, 2018.

Copyright Notice

Copyright (c) 2017 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. Carrying PCE Discovery Information in BGP	3
2.1. PCE NLRI	3
2.1.1. PCE Descriptors	4
2.2. PCE Attribute TLVs	5
2.2.1. PCE Domain TLV	6
2.2.2. Neighbor PCE Domain TLV	6
3. Operational Considerations	7
4. IANA Considerations	7
5. Security Considerations	7
6. Contributors	7
7. Acknowledgements	8
8. References	8
8.1. Normative References	8
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

In networks where a Path Computation Element (PCE) is used for path computation, it is desirable for the Path Computation Clients (PCCs) to discover dynamically and automatically a set of PCEs along with certain information relevant for PCE selection. [RFC5088] and [RFC5089] define the PCE discovery mechanisms based on Interior Gateway Protocols (IGP). When PCCs are LSRs participating in the IGP (OSPF or IS-IS), and PCEs are either LSRs or servers also participating in the IGP, an effective mechanism for PCE discovery within an IGP routing domain consists of utilizing IGP advertisements.

[RFC4674] presents a set of requirements for a PCE discovery mechanism. This includes the discovery by a PCC of a set of one or more PCEs which may potentially be in some other domains. This is a desirable function in the case of inter-domain path computation. For example, Backward Recursive Path Computation (BRPC) [RFC5441] can be used by cooperating PCEs to compute an inter-AS path, in which case the discovery of PCE as well as the domain information is useful.

BGP has been extended for north-bound distribution of routing and TE information to PCE [RFC7752] and [I-D.ietf-idr-te-pm-bgp]. Similarly this document extends BGP to also carry the PCE discovery information.

This document defines extensions to BGP to allow a PCE to advertise its location, along with some information useful to a PCC for the PCE selection, so as to satisfy dynamic PCE discovery requirements set forth in [RFC4674].

This specification contains two parts: definition of a new BGP-LS NLRI [RFC7752] that describes PCE information and definition of PCE Attribute TLVs as part of BGP-LS attributes.

2. Carrying PCE Discovery Information in BGP

2.1. PCE NLRI

The PCE discovery information is advertised in BGP UPDATE messages using the MP_REACH_NLRI and MP_UNREACH_NLRI attributes [RFC4760]. The "Link- State NLRI" defined in [RFC7752] is extended to carry the PCE information. BGP speakers that wish to exchange PCE discovery information MUST use the BGP Multiprotocol Extensions Capability Code (1) to advertise the corresponding (AFI, SAFI) pair, as specified in [RFC4760].

The format of "Link-State NLRI" is defined in [RFC7752]. A new "NLRI Type" is defined for PCE Information as following:

- o Type = TBD1: PCE NLRI

The format of PCE NLRI is shown in the following figure:

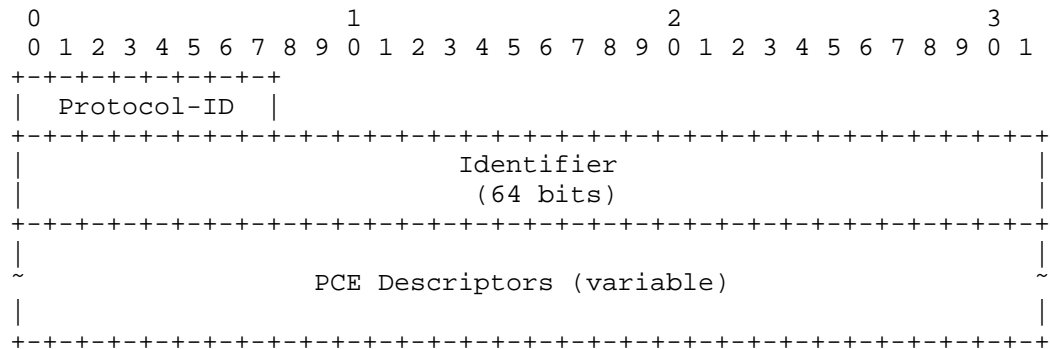


Figure 1. PCE NLRI

The 'Protocol-ID' field is defined in [RFC7752], to be set to the appropriate value that indicates the source of the PCE information. If BGP speaker and PCE are co-located, the Protocol-ID SHOULD be set to "Direct". If PCE information to advertise is configured at the BGP speaker, the Protocol-ID SHOULD be set to "Static configuration".

As defined in [RFC7752], the 64-Bit 'Identifier' field is used to identify the "routing universe" where the PCE belongs.

2.1.1.1. PCE Descriptors

The PCE Descriptor field is a set of Type/Length/Value (TLV) triplets. The format of each TLV is as per Section 3.1 of [RFC7752]. The PCE Descriptor TLVs uniquely identify a PCE. The following PCE descriptor are defined -

Codepoint	Descriptor TLV	Length
TBD2	IPv4 PCE Address	4
TBD3	IPv6 PCE Address	16

Table 1: PCE Descriptors

The PCE address TLVs specifies an IP address that can be used to reach the PCE. The PCE-ADDRESS Sub-TLV defined in [RFC5088] and [RFC5089] is used in the OSPF and IS-IS respectively. The format of the PCE address TLV are -

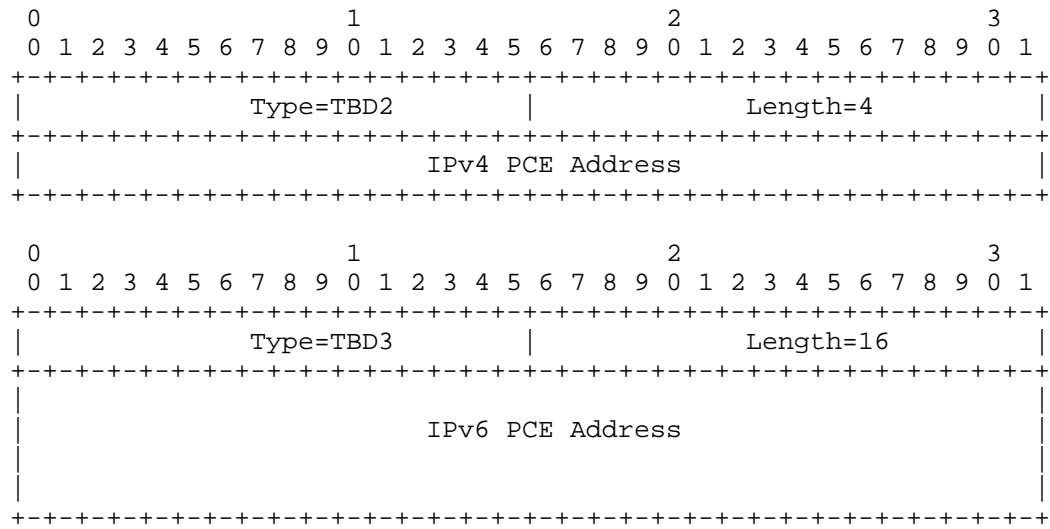


Figure 2. PCE Address TLVs

When the PCE has both an IPv4 and IPv6 address, both the TLVs MAY be included.

2.2. PCE Attribute TLVs

PCE Attribute TLVs are TLVs that may be encoded in the BGP-LS attribute [RFC7752] with a PCE NLRI. The format of each TLV is as per Section 3.1 of [RFC7752]. The format and semantics of the Value fields in some PCE Attribute TLVs correspond to the format and semantics of the Value fields in IS-IS PCED Sub-TLV, defined in [RFC5089]. Other PCE Attribute TLVs are defined in this document.

The following PCE Attribute TLVs are valid in the BGP-LS attribute with a PCE NLRI:

TLV Code Point	Description	IS-IS TLV /Sub-TLV	Reference (RFC/Section)
TBD4	Path Scope	5/2	[RFC5089]/4.2
TBD5	PCE Domain	-	-
TBD6	Neighbor PCE Domain	-	-
TBD7	PCE Capability	5/5	[RFC5089]/4.5

Table 2: PCE Attribute TLVs

The format and semantics of Path Scope and PCE capability is as per [RFC5089]. The Path Scope TLV is mandatory.

2.2.1. PCE Domain TLV

The PCE Domain TLV specifies a PCE-Domain (IGP area and/or AS) where the PCE has topology visibility and through which the PCE can compute paths.

```

      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=TBD5                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Length                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Domain Sub-TLVs (variable)                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The length of this TLV is variable. The value contains one or more domain sub-TLVs as listed below -

Sub-TLV Code Point	Description	Length
512	Autonomous System	4
514	OSPF Area-ID	4
1027	IS-IS Area Identifier	Variable

Multiple sub-TLVs MAY be included, when the PCE has visibility into multiple PCE-Domains.

2.2.2. Neighbor PCE Domain TLV

The Neighbor PCE Domain TLV specifies a neighbor PCE-Domain (IGP area and/or AS) toward which a PCE can compute paths.

```

      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=TBD6                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Length                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Domain Sub-TLVs (variable)                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

The length of this TLV is variable. The value contains one or more domain sub-TLVs as listed above. Multiple sub-TLVs MAY be included, when the PCE can compute paths towards several neighbor PCE-Domains.

3. Operational Considerations

Existing BGP-LS operational procedures apply to the advertisement of PCE information as per [RFC7752]. This information is treated as pure application level data which has no immediate impact on forwarding states. The PCE information SHOULD be advertised only to the domains where such information is allowed to be used. This can be achieved by policy control on the ASBRs.

The PCE information is considered relatively stable and does not change frequently, thus this information will not bring significant impact on the amount of BGP updates in the network.

4. IANA Considerations

IANA needs to assign a new NLRI Type for 'PCE NLRI' from the "BGP-LS NLRI-Types" registry.

IANA needs to assign new TLV code point as per Table 1 and 2 from the "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs" registry.

[Editor's Note - Check if name of the registry should be changes with following instructions - Further IANA is requested to rename the registry as "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, PCE Descriptor, and Attribute TLVs".]

5. Security Considerations

Procedures and protocol extensions defined in this document do not affect the BGP security model. See the 'Security Considerations' section of [RFC4271] for a discussion of BGP security. Also refer to [RFC4272] and [RFC6952] for analysis of security issues for BGP.

Existing BGP-LS security considerations as per [RFC7752] continue to apply.

6. Contributors

The following individuals gave significant contributions to this document:

Takuya Miyasaka
KDDI Corporation
ta-miyasaka@kddi.com

7. Acknowledgements

The authors would like to thank Zhenbin Li, Hannes Gredler, Jan Medved, Adrian Farrel, Julien Meuric and Jonathan Hardwick for the valuable discussion and comments.

8. References

8.1. 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>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <<http://www.rfc-editor.org/info/rfc4760>>.
- [RFC5088] Le Roux, JL., Ed., Vasseur, JP., Ed., Ikejiri, Y., and R. Zhang, "OSPF Protocol Extensions for Path Computation Element (PCE) Discovery", RFC 5088, DOI 10.17487/RFC5088, January 2008, <<http://www.rfc-editor.org/info/rfc5088>>.
- [RFC5089] Le Roux, JL., Ed., Vasseur, JP., Ed., Ikejiri, Y., and R. Zhang, "IS-IS Protocol Extensions for Path Computation Element (PCE) Discovery", RFC 5089, DOI 10.17487/RFC5089, January 2008, <<http://www.rfc-editor.org/info/rfc5089>>.
- [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, <<http://www.rfc-editor.org/info/rfc7752>>.

8.2. Informative References

- [I-D.ietf-idr-te-pm-bgp]
Previdi, S., Wu, Q., Gredler, H., Ray, S.,
jeffrant@gmail.com, j., Filsfils, C., and L. Ginsberg,
"BGP-LS Advertisement of IGP Traffic Engineering
Performance Metric Extensions", draft-ietf-idr-te-pm-
bgp-06 (work in progress), June 2017.
- [RFC4272] Murphy, S., "BGP Security Vulnerabilities Analysis",
RFC 4272, DOI 10.17487/RFC4272, January 2006,
<<http://www.rfc-editor.org/info/rfc4272>>.
- [RFC4674] Le Roux, J., Ed., "Requirements for Path Computation
Element (PCE) Discovery", RFC 4674, DOI 10.17487/RFC4674,
October 2006, <<http://www.rfc-editor.org/info/rfc4674>>.
- [RFC5441] Vasseur, JP., Ed., Zhang, R., Bitar, N., and JL. Le Roux,
"A Backward-Recursive PCE-Based Computation (BRPC)
Procedure to Compute Shortest Constrained Inter-Domain
Traffic Engineering Label Switched Paths", RFC 5441,
DOI 10.17487/RFC5441, April 2009,
<<http://www.rfc-editor.org/info/rfc5441>>.
- [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,
<<http://www.rfc-editor.org/info/rfc6952>>.

Authors' Addresses

Jie Dong
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: jie.dong@huawei.com

Mach(Guoyi) Chen
Huawei Technologies
Huawei Campus, No. 156 Beiqing Rd.
Beijing 100095
China

Email: mach.chen@huawei.com

Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore, Karnataka 560066
India

Email: dhruv.ietf@gmail.com

Jeff Tantsura
Individual
US

Email: jefftant.ietf@gmail.com

Kenji Kumaki
KDDI Corporation
Garden Air Tower, Iidabashi, Chiyoda-ku
Tokyo 102-8460
Japan

Email: ke-kumaki@kddi.com

Tomoki Murai
Furukawa Network Solution Corp.
5-1-9, Higashi-Yawata, Hiratsuka
Kanagawa 254-0016
Japan

Email: murai@fnsc.co.jp

IDR Working Group
Internet-Draft
Intended status: Standards Track
Expires: February 21, 2020

G. Van de Velde, Ed.
W. Henderickx
M. Bocci
Nokia
K. Patel
Arrcus
August 20, 2019

Signalling ERLD using BGP-LS
draft-ietf-idr-bgp-ls-segment-routing-rld-05

Abstract

This document defines the attribute encoding to use for BGP-LS to expose ERLD "Entropy capable Readable Label Depth" from a node to a centralised controller (PCE/SDN).

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 [1].

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 21, 2020.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction	2
2. Conventions used in this document	2
2.1. Terminology	3
3. Problem Statement	3
4. Advertising of ERLD in BGP-LS	3
5. Security Considerations	4
6. Acknowledgements	4
7. IANA Considerations	4
8. References	4
8.1. Normative References	4
8.2. Informative References	5
Authors' Addresses	5

1. Introduction

When Segment Routing tunnels are computed by a centralised controller, it is beneficial that the controller knows the ERLD (Entropy Readable Label Depth) of each node or link a tunnel traverses. A network node signalling an ERLD MUST support the ability to read the signalled number of labels before any action is done upon the packet and SHOULD support entropy awareness found within the signalled ERLD depth.

ERLD awareness of each node will allow a network SDN controller to influence the path used for each tunnel. The SDN controller may for example only create tunnels with a label stack smaller or equal as the ERLD of each node on the path. This will allow the network to behave accordingly (e.g. make use of Entropy Labels to improve ECMP) upon the imposed Segment Routing label stack on each packet.

This document describes how to use BGP-LS to expose the ERLD of a node.

2. Conventions used in this document

2.1. Terminology

BGP-LS: Distribution of Link-State and TE Information using Border Gateway Protocol

ERLD: Entropy capable Readable Label Depth

ELC: Entropy Label Capability

MSD: Maximum SID Depth

SID: Segment Identifier

3. Problem Statement

For Segment Routing technology both ISIS [7] and OSPF [6] have proposed extensions to signal the ERLD (Entropy Readable Label Depth) and ELC (Entropy Label Capability) of a node. However, if a network SDN controller is connected to the network through a BGP-LS session and not through either ISIS or OSPF technology, then both ERLD and ELC needs to be signalled using BGP-LS encoding. This document describes the extension BGP-LS requires to signal ERLD.

A network SDN controller having awareness of the ERLD can for example use it as a constraint on path computation to make sure that high bandwidth LSPs are not placed on LAG (Link Aggregation Group), containing links with smaller member bandwidth, if they know the entropy label cannot be processed by the node at the ingress to the link.

4. Advertising of ERLD in BGP-LS

Both ISIS [7] and OSPF [6] have proposed extensions to signal the ERLD (Entropy Readable Label Depth) and ELC (Entropy Label Capability) using new MSD-type of the Node MSD sub-Type TLV RFC8491 [4] or RFC8476 [3].

This document defines a new node BGP MSD sub-type TLV from draft-ietf-idr-bgp-ls-segment-routing-msd [5] to signal the ERLD.

A BGP-LS router exporting the IGP LSDB, MUST NOT encode the IGP ERLD value in an BGP-LS ERLD attribute, if the associated ELC is not signalled.

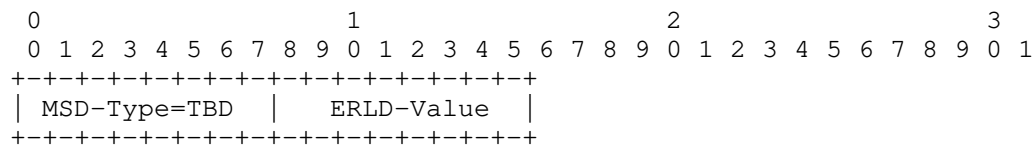


Figure 1

The BGP-LS ERLD is encoded as a node MSD sub-type defined in the IANA registry titled "IGP MSD-Types" under the "Interior Gateway Protocol (IGP) Parameters" registry created by RFC8491 [4].

The ERLD-Value field in the range between 0 to 255 is set to the BGP-LS imported IGP ERLD. The value of 0 represents lack of ability to read a label stack of any depth, any other value represents the readable label depth of the node.

5. Security Considerations

This document does not introduce security issues beyond those discussed in RFC7752 [2]

6. Acknowledgements

Thanks to discussions with Acee Lindem, Jeff Tantsura, Stephane Litkowski, Bruno Decraene, Kireeti Kompella, John E. Drake and Carlos Pignataro to bring the concept of combining ELC and RLD into a single ERLD signalled parameter more suitable for SDN controller based networks.

7. IANA Considerations

This document requests assigning new BGP MSD sub-TLV code-points as described in section 4.

Note: placeholder IANA request

8. References

8.1. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <<http://xml.resource.org/public/rfc/html/rfc2119.html>>.

- [2] 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>>.
- [3] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", RFC 8476, DOI 10.17487/RFC8476, December 2018, <<https://www.rfc-editor.org/info/rfc8476>>.
- [4] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", RFC 8491, DOI 10.17487/RFC8491, November 2018, <<https://www.rfc-editor.org/info/rfc8491>>.
- [5] Tantsura, J., Chunduri, U., Talaulikar, K., Mirsky, G., and N. Triantafyllis, "draft-ietf-idr-bgp-ls-segment-routing-msd", June 2019.

8.2. Informative References

- [6] Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "draft-ietf-ospf-mpls-elc", May 2019.
- [7] Xu, X., Kini, S., Sivabalan, S., Filsfils, C., and S. Litkowski, "draft-ietf-isis-mpls-elc", May 2019.

Authors' Addresses

Gunter Van de Velde (editor)
Nokia
Antwerp
BE

Email: gunter.van_de_velde@nokia.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Matthew Bocci
Nokia
Shoppenhangers Road
Maidenhead, Berks
UK

Email: matthew.bocci@nokia.com

Keyur Patel
Arrcus
USA

Email: keyur@arrcus.com

IDR
Internet-Draft
Updates: 4271, 4360, 7153 (if approved)
Intended status: Standards Track
Expires: September 4, 2018

Z. Li
China Mobile
J. Dong
Huawei Technologies
March 3, 2018

Carry congestion status in BGP community
draft-li-idr-congestion-status-extended-community-07

Abstract

To aid BGP receiver to steer the AS-outgoing traffic among the exit links, this document introduces a new BGP community, congestion status community, to carry the link bandwidth and utilization information, especially for the exit links of one AS. If accepted, this document will update RFC4271, RFC4360 and RFC7153.

The introduced congestion status community is not used to impact the decision process of BGP specified in section 9.1 of RFC4271, but can be used by route policy to impact the data forwarding behavior.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 4, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

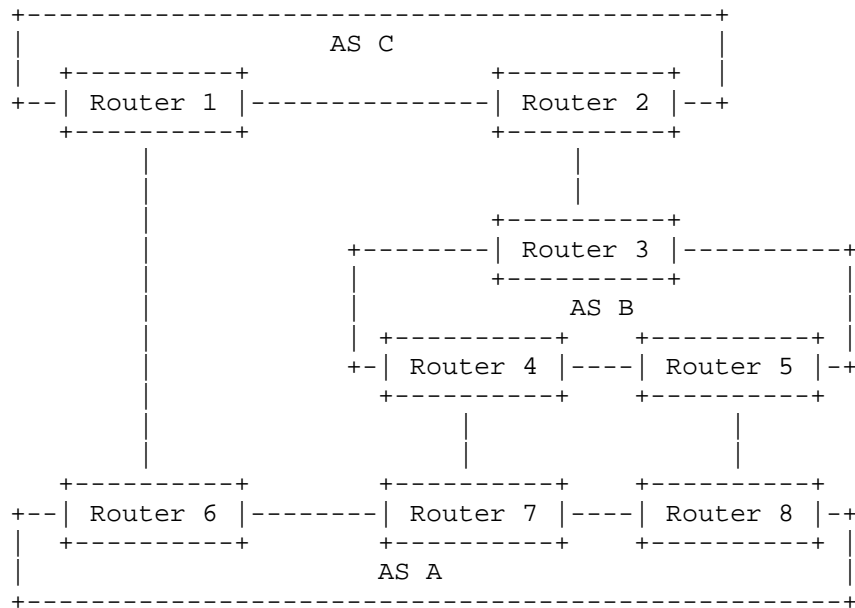
Table of Contents

1. Introduction	2
2. Requirements Language	4
3. Previous Work	4
4. Solution Alternative 1: Extended Community	4
5. Solution Alternative 2: Large Community	6
6. Solution Alternative 3: Community Container	6
7. Deployment Considerations	8
8. Security Considerations	9
9. IANA Considerations	9
10. Acknowledgments	9
11. References	9
11.1. Normative References	10
11.2. Informative References	10
Appendix A. Bandwidth Values	11
Authors' Addresses	12

1. Introduction

Knowing the congestion status (bandwidth and utilization) of the AS exit links is useful for traffic steering, especially for steering the AS outgoing traffic among the exit links. Section 7 of [I-D.gredler-idr-bgplu-epe] explicitly specifies this kind of requirement, which is also needed in our field network.

The following figure is used to illustrate the benefits of knowing the congestion status of the AS exit links. AS A has multiple exit links connected to AS B. Both AS A and B has exit link to AS C, and AS B provides transit service for AS A. Due to cost or some other reasons, AS A prefers using AS B to transmit its' traffic to AS C, not the directly connected link between AS A and C. If the exit routers, Router 7 and 8, in AS A tell their iBGP peers the congestion status of the exit links, the peers in turn can steer some outgoing traffic toward the less loaded exit link. If AS A knows the link between AS B and AS C is congested, it can steer some traffic towards AS C from AS B to the directly connected link by applying some route policies.



This document introduces new BGP extensions to deliver the congestion status of the exit link to other BGP speakers. The BGP receiver can then use this community to deploy route policy, thus steer AS outgoing traffic according to the congestion status of the exit links. This mechanism can be used by both iBGP and eBGP.

In this version, we provide three solution alternatives according to the discussion in the face to face meetings and mail list. After adoption, one solution will be selected as the final solution based on the working group consensus.

In a network deployed SDN (Software Defined Network) controller, congestion status extended community can be used by the controller to steer the AS outgoing traffic among all the exit links from the perspective of the whole network.

For the network with Route Reflectors (RRs) [RFC4456], RRs by default only advertise the best route for a specific prefix to their clients. Thus RR clients has no opportunity to compare the congestion status among all the exit links. In this situation, to allow RR clients learning all the routes for a specific prefix from all the exit links, RRs are RECOMMENDED to enable add-path functionality [RFC7911].

To emphasize, the introduced new BGP extensions have no impact on the decision process of BGP specified in section 9.1 of [RFC4271], but can be used by route policy to impact the data forwarding behavior.

2. Requirements Language

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

3. Previous Work

In [constrained-multiple-path], authors from France Telecom also specified the requirement to know the congestion status of a link.

To aid a router to perform unequal cost load balancing, experts from Cisco introduced Link Bandwidth Extended Community in [link-bandwidth-community] to carry the cost to reach the external BGP neighbor. The cost can be either configured per neighbor or derived from the bandwidth of the link that connects the router to a directly connected external neighbor. This document was accepted by the IDR working group, but expired in 2013.

Link Bandwidth Extended Community only carries the link bandwidth of the exit link. The method provided in our document can carry the link bandwidth together with the link utilization information. What the BGP receiver needs to impact its traffic steering policy is the up-to-date unused link bandwidth, which can be derived from the link bandwidth and link utilization. Since Link Bandwidth Extended Community is expired, the BGP speaker who receives update message with both Link Bandwidth Extended Community and Congestion Status Community SHOULD ignore the Link Bandwidth Extended Community and use the Congestion Status Community.

4. Solution Alternative 1: Extended Community

As described in [RFC4360], the extended community attribute is an 8-octet value with the first one or two octets to indicate the type of this attribute. Since congestion status community needs to be delivered from one AS to other ASes, and used by the BGP speakers both in other ASes and within the same AS as the sender, it MUST be a transitive extended community, i.e. the T bit in the first octet MUST be zero.

We only define the congestion status community for four-octet AS number [RFC6793], since all the BGP speakers can handle four-octet AS number now and the two-octet AS numbers can be mapped to four-octet

AS numbers by setting the two high-order octets of the four-octet field to zero, as per [RFC6793].

Congestion status community is a sub-type allocated from Transitive Four-Octet AS-Specific Extended Community Sub-Types defined in section 5.2.4 of [RFC7153]. Its format is as Figure 1.

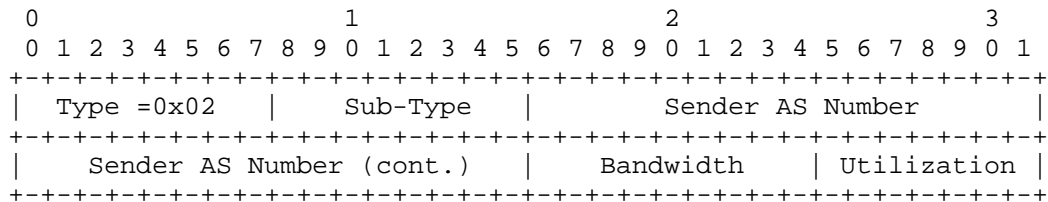


Figure 1: Congestion status extended community

Type: 1 octet. This field MUST be 0x02 to indicate this is a Transitive Four-Octet AS-Specific Extended Community.

Sub-Type: 1 octet. It is used to indicate this is a Congestion Status Extended Community. Its value is to be assigned by IANA.

Sender AS Number: 4 octets. Its value is the AS number of the BGP speaker who generates this congestion status extended community. If the generator has 2-octet AS number, it MUST encode its AS number in the last (low order) two bytes and set the first (high order) two bytes to zero, as per [RFC6793].

Bandwidth: 1 octet. Its value is the bandwidth of the exit link in unit of 10 gbps (gigabits per second). The link with bandwidth less than 10 gbps is not suitable to use this feature. To reflect the practice that sometimes the traffic is rate limited to a capacity smaller than the physical link, the value of the bandwidth can be the configured capacity of the link. The available configured capacity can be calculated from this field together with Utilization field. Zero means the bandwidth is unknown or is not advertised to other peers.

Utilization: 1 octet. Its value is the utilization of the exit link in unit of percent. A value bigger than 100 means the incoming traffic is higher than the link capacity. We can use the "Utilization" field together with the "Bandwidth" field to calculate the traffic load that we can further steer to this exit link.

5. Solution Alternative 2: Large Community

As described in [RFC8092], the BGP large community attribute is an optional transitive path attribute of variable length, consisting of 12-octet values. The BGP large community attribute is mainly used to extend the size of BGP Community [RFC1997] and Extended Community [RFC4360], thus to accommodate at least two four-octet ASNs [RFC6793]. As shown in the following figure, the format of the 12-octet BGP Large Community value is not suitable to be used to define new type for congestion status community.

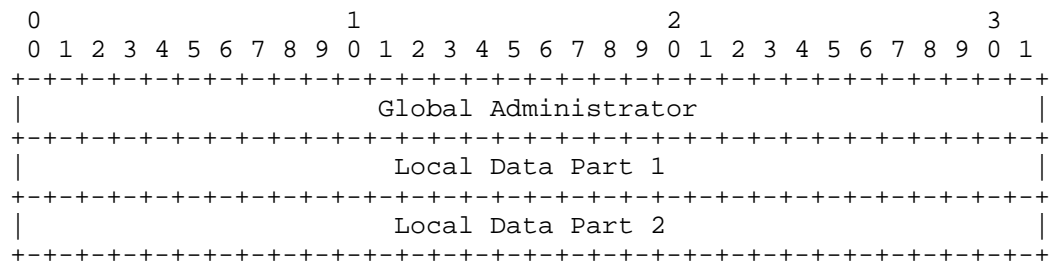


Figure 2

Global Administrator: A four-octet namespace identifier.

Local Data Part 1: A four-octet operator-defined value.

Local Data Part 2: A four-octet operator-defined value.

6. Solution Alternative 3: Community Container

As described in [I-D.ietf-idr-wide-bgp-communities], the BGP Community Container has flexible encoding format, which we can use to define the congestion status community.

A new type of the BGP Community Container is defined for the congestion status community, which has the same common header as the BGP Community Container with the following encoding format.

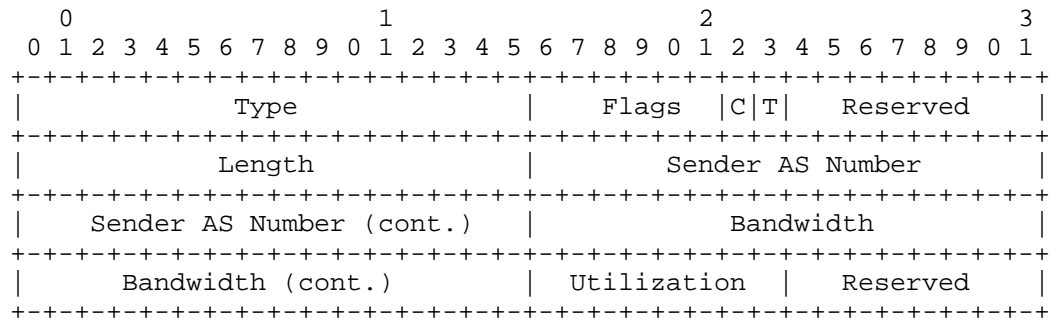


Figure 3

Type: 2 octets. Its value is to be assigned by IANA from the registry "BGP Community Container Types" to indicate this is the Congestion Status Community.

Flags: 1 octet. C and T bits MUST be set to indicate the Congestion Status Community is transitive across confederation and AS boundaries. The other bits in Flags field MUST be set to zero when originated and SHOULD be ignored upon receipt.

Reserved: Reserved fields are reserved for future definition, which MUST be set to zero when originated and SHOULD be ignored upon receipt.

Length: 2 octets. This field represents the total length of a given container's contents in octets.

Sender AS Number: 4 octets. Its value is the AS number of the BGP speaker who generates this congestion status community. If the generator has 2-octet AS number, it MUST encode its AS number in the last (low order) two bytes and set the first (high order) two bytes to zero, as per [RFC6793].

Bandwidth: 4 octets. Its value is the bandwidth of the exit link in IEEE floating point format (see [IEEE.754.1985]), expressed in bytes per second. Zero means the bandwidth is unknown or is not advertised to other peers. Appendix A lists some typical bandwidth values, most of which are extracted from Section 3.1.2 of [RFC3471].

To reflect the practice that sometimes the traffic is rate limited to a capacity smaller than the physical link, the value of the bandwidth can be the configured capacity of the link. The available configured capacity can be calculated from this field together with Utilization field.

Utilization: 1 octet. Its value is the utilization of the exit link in unit of percent. A value bigger than 100 means the incoming traffic is higher than the link capacity. We can use the "Utilization" field together with the "Bandwidth" field to calculate the traffic load that we can further steer to this exit link.

7. Deployment Considerations

o To avoid route oscillation

The exit router SHOULD set a threshold. When the utilization change reaches the threshold, the exit router SHOULD generate a BGP update message with congestion status community.

Implementations SHOULD further reduce the BGP update messages triggered by link utilization change using the method similar to BGP Route Flap Damping [RFC2439]. When link utilization change by small amounts that fall under thresholds that would cause the announcement of BGP update message, implementations SHOULD suppress the announcement and set the penalty value accordingly.

To reduce the update churn introduced, when one BGP router needs to re-advertise a BGP path due to attribute changes, it SHOULD update its Congestion Status Community at the same time. Supposing there are N ASes on the way from the far end egress BGP speaker to the final ingress BGP speaker, this allows reducing the update churn as the final ingress BGP speaker will receive a single UPDATE refreshing the N communities, rather than N UPDATES, each refreshing one community.

o To avoid traffic oscillation

Traffic oscillation means more traffic than expected is attracted to the low utilized link, and some traffic has to be steered back to other links.

Route policy is RECOMMENDED to be set at the exit router. Congestion status community is only conveyed for some specific routes or only for some specific BGP peers.

Congestion status community can also be used in a SDN network. The SDN controller uses the exit link utilization information to steer the Internet access traffic among all the exit links from the perspective of the whole network.

o Other Consens

To avoid forwarding loops incremental deployment issues, complications in error handling, the reception of such community over IBGP session SHOULD NOT influence routing decision unless tunneling is used to reach the BGP Next-Hop.

8. Security Considerations

This document defines a new BGP community to carry the congestion status of the exit link. It is up to the BGP receiver to trust the congestion status communities or not. Following deployment models can be considered.

The BGP receiver may choose to only trust the congestion status communities generated by some specific ASes or containing bandwidth greater than a specific value.

You can filter the congestion status communities at the border of your trust/administrative domain. Hence all the ones you receive are trusted.

You can record the communities received over time, monitor the congestion e.g. via probing, detect inconsistency and choose to not trust anymore the ASes which advertise fake news.

9. IANA Considerations

For solution alternative 1, one sub-type is solicited to be assigned from Transitive Four-Octet AS-Specific Extended Community Sub-Types registry to indicate the Congestion Status Community defined in this document.

For solution alternative 3, one community value is solicited to be assigned from the registry "Registered Type 1 BGP Wide Community Community Types" to indicate the Congestion Status Community defined in this document.

10. Acknowledgments

We appreciate the constructive suggestions received from Bruno Decraene. Many thanks to Rudiger Volk, Susan Hares, John Scudder, Randy Bush for their review and comments to improve this document.

11. References

11.1. Normative References

- [I-D.ietf-idr-wide-bgp-communities]
Raszuk, R., Haas, J., Lange, A., Decraene, B., Amante, S.,
and P. Jakma, "BGP Community Container Attribute", draft-
ietf-idr-wide-bgp-communities-04 (work in progress), March
2017.
- [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>>.
- [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>>.
- [RFC8092] Heitz, J., Ed., Snijders, J., Ed., Patel, K., Bagdonas,
I., and N. Hilliard, "BGP Large Communities Attribute",
RFC 8092, DOI 10.17487/RFC8092, February 2017,
<<https://www.rfc-editor.org/info/rfc8092>>.

11.2. Informative References

- [constrained-multiple-path]
Boucadair, M. and C. Jacquenet, "Constrained Multiple BGP
Paths", October 2010, <[https://www.ietf.org/archive/id/
draft-boucadair-idr-constrained-multiple-path-00.txt](https://www.ietf.org/archive/id/draft-boucadair-idr-constrained-multiple-path-00.txt)>.
- [I-D.gredler-idr-bgplu-epe]
Gredler, H., Vairavakkalai, K., R, C., Rajagopalan, B.,
Aries, E., and L. Fang, "Egress Peer Engineering using
BGP-LU", draft-gredler-idr-bgplu-epe-11 (work in
progress), October 2017.

[link-bandwidth-community]

Mohapatra, P. and R. Fernando, "BGP Link Bandwidth Extended Community", January 2013, <<https://www.ietf.org/archive/id/draft-ietf-idr-link-bandwidth-06.txt>>.

[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>>.

[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>>.

[RFC3471] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description", RFC 3471, DOI 10.17487/RFC3471, January 2003, <<https://www.rfc-editor.org/info/rfc3471>>.

[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>>.

[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>>.

[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. Bandwidth Values

Some typical bandwidth values encoded in 32-bit IEEE floating point format are enumerated below.

Link Type	Bit-rate (Mbps)	Bandwidth Value (Bytes/Sec) (32-bit IEEE Floating point)
-----	-----	-----
E1	2.048	0x487A0000
Ethernet	10.00	0x49989680
Fast Ethernet	100.00	0x4B3EBC20
OC-3/STM-1	155.52	0x4B9450C0
OC-12/STM-4	622.08	0x4C9450C0
GigE	1000.00	0x4CEE6B28
OC-48/STM-16	2488.32	0x4D9450C0
OC-192/STM-64	9953.28	0x4E9450C0
10GigE	10000.00	0x4E9502F9
OC-768/STM-256	39813.12	0x4F9450C0
100GigE	100000.00	0x503A43B7

Authors' Addresses

Zhenqiang Li
China Mobile
No.32 Xuanwumenxi Ave., Xicheng District
Beijing 100032
P.R. China

Email: li_zhenqiang@hotmail.com

Jie Dong
Huawei Technologies
Huawei Campus, No.156 Beiqing Rd.
Beijing 100095
P.R. China

Email: jie.dong@huawei.com

Interdomain Routing
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2018

K. Patel, Ed.
Arrcus
M. Jethanandani, Ed.

S. Hares, Ed.
Hickory Hill Consulting
November 15, 2017

BGP YANG Model
draft-mks-idr-bgp-yang-model-01

Abstract

This Internet draft provides a set of example text for the replacement of draft-ietf-idr-bgp-model-02.txt with the IETF models based on the Network Management Datastore Architecture to be released in draft-ietf-idr-bgp-model-03.txt. This draft is provided for the IDR WG by potential editors as example text for draft-ietf-idr-bgp-model-03.txt for the yang models. Please send review comments or suggestions to these potential editors and the IDR working group (idr@ietf.org).

This document defines a YANG data model for configuring and managing BGP, including protocol, policy, and operational aspects 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 May 19, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	2
1.1. Goals and approach	3
2. Model overview	4
2.1. BGP protocol configuration	5
2.2. Policy configuration overview	7
2.3. Operational state overview	8
3. Relation to other YANG data models	8
4. Security Considerations	9
5. IANA Considerations	9
6. YANG modules	9
7. BGP main module and submodule for base items	10
8. BGP types	50
9. BGP policy data	59
10. References	73
10.1. Normative references	73
10.2. Informative references	74
Appendix A. Acknowledgements	75
Appendix B. Change summary	75
B.1. Changes between revisions -01 and -02	75
B.2. Changes between revisions -00 and -01	75
Authors' Addresses	75

1. Introduction

This Internet draft is a set of example text for the replacement of draft-ietf-idr-bgp-model-02.txt with a version of the yang models which are compatible with the Network Management Datastore Architecture to be released in draft-ietf-idr-bgp-model-03.txt. This draft is only for provided an example structure for the IDR WG by the potential editors for draft-ietf-idr-bgp-model-03.txt. The authors of the draft-ietf-bgp-model-02.txt are:

- o Anees Shaikh
- o Rob Shakir
- o Keyur Patel
- o Susan Hares
- o Kevin D'Souza
- o Deepak Bansal
- o Alex Clemm
- o Alex Zhdankin
- o Mahesh Jethanandani
- o Xufeng Liu

This document describes a YANG data model for the BGP [RFC4271] protocol, including various protocol extensions, policy configuration, as well as defining key operational state data. 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.

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 [RFC4271], [RFC1997], [RFC4456], [RFC4760], [RFC3065], [RFC2439], [RFC4724], and [RFC6811].

Along with configuration of base BGP features, this model also addresses policy configuration, by providing "hooks" for applying policies, and also defining BGP-specific policy features. The BGP policy features are intended to be used with the general routing policy model defined in [I-D.ietf-rtgwg-policy-model]. The model also supports operational state data to provide a common model for reading BGP-related state from a BGP speaker.

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

- o The global BGP instance, and neighbors whose configuration is specified individually, or templated with the use of peer-groups.
- o The address families that are supported by peers, and the global configuration which relates to them.
- o The policy configuration "hooks" and BGP-specific policy features that relate to a neighbor - controlling the import and export of NLRI's.

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.

2. Model overview

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

- o base protocol configuration -- configuration affecting BGP protocol-related operations, defined at various levels of hierarchy.
- o multiprotocol configuration -- configuration affecting individual address-families within BGP [RFC4760].
- o neighbor configuration -- configuration affecting an individual neighbor within BGP.

- o neighbor multiprotocol configuration -- configuration affecting individual address-families for a neighbor within BGP.
- o policy configuration -- hooks for application of the policies defined in [I-D.ietf-rtgwg-policy-model] that act on routes sent (received) to (from) peers or other routing protocols and BGP-specific policy features.
- o operational state -- variables used for monitoring and management of BGP operations.

These modules also make use of standard Internet types, such as IP addresses and prefixes, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

Throughout the model, the approach described in NMDA [I-D.ietf-netmod-revised-datastores] is used to represent running configuration, intended and operational datastore. That is to say, that the model defines a single container, and it is the implementation of the different datastores that reflects the value of a given node in either the <running>, <intended> or <operational> datastore.

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.

```

+--rw bgp!
  +--rw global
  |   +-- (global-configuration-options)
  +--rw neighbors
  |   +--rw neighbor* [neighbor-address]
  |   +-- (neighbor-configuration-options)
  +--rw peer-groups
  |   +--rw peer-group* [peer-group-name]
  |   +-- (neighbor-configuration-options)
```

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 following address-families are currently supported by the model:

```

+--rw bgp!
  +--rw global
    +--rw afi-safis
      +--rw afi-safi* [afi-safi-name]
        +--rw afi-safi-name    -> ../config/afi-safi-name
        |
        +--rw ipv4-unicast
        |   ...
        +--rw ipv6-unicast
        |   ...
        +--rw ipv4-labelled-unicast
        |   ...
        +--rw ipv6-labelled-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 references the generic YANG routing policy model described in [I-D.ietf-rtgwg-policy-model], which represents a condition-action policy framework for routing. This model adds BGP-specific conditions (e.g., matching on the community attribute), and actions (e.g., setting local preference) to the generic policy framework.

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

- o within the global instance, where a policy applies to all address-families for all peers.
- o on a global AFI-SAFI basis, where policies apply to all peers for a particular address-family.
- o on a per-peer-group or per-neighbor basis - where the policy applies to all address-families for the particular group or neighbor.

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

```

+--rw bgp
  +--rw global
    |   +--rw afi-safi
    |   |   +--rw afi-safi* [afi-safi-name]
    |   |   +--rw apply-policy
    |   +--rw apply-policy
  +--rw neighbors
    |   +--rw neighbor* [neighbor-address]
    |   |   +--rw afi-safi
    |   |   |   +--rw afi-safi* [afi-safi-name]
    |   |   |   +--rw apply-policy
    |   |   +--rw apply-policy
  +--rw peer-groups
    |   +--rw peer-group* [peer-group-name]
    |   |   +--rw afi-safi
    |   |   |   +--rw afi-safi* [afi-safi-name]
    |   |   |   +--rw apply-policy
    |   |   +--rw apply-policy

```

2.3. Operational state overview

The BGP operational model contains data which relates to the operational state of the various elements of the BGP router. As noted in Section 2 - the approach described in NMDA [I-D.ietf-netmod-revised-datastores] is utilized for the modeling of operational and statistical data. To this end, the "-state" groupings (those that contain derived operational parameters) is not a separate container, but is instead collapsed into one container that defines both the read-write and read-only nodes. In some cases, operational information may be relevant to one instance of a common grouping, but not another - for example, the number of received, advertised, and installed prefixes is relevant on a per-neighbor-basis, but is not required (or meaningful) in the peer-group context. Groupings are defined with the appropriate operational state data accordingly.

3. Relation to other YANG data models

The BGP model is intended to work within a larger framework model, such as the Network Instance model [I-D.ietf-rtgwg-ni-model] which provides a comprehensive model for defining VRFs, associated routing protocols, multiple protocol instances, and inter-protocol and inter-instance routing policies. The current version of the model imports

and instantiates the BGP model in its tree at /network-instances/network-instance/protocols/protocol/bgp/...

It is also possible to integrate the BGP model with the Routing Management model [I-D.ietf-netmod-routing-cfg] or the Network Device Organizational Model [I-D.rtgyangdt-rtgwg-device-model], both of which define the notion of routing instances, or VRFs.

4. Security Considerations

BGP configuration has a significant impact on network operations, and as such any related protocol or model carries potential security risks.

YANG data models are generally designed to be used with the NETCONF protocol over an SSH transport. This provides an authenticated and secure channel over which to transfer BGP configuration and operational data. Note that use of alternate transport or data encoding (e.g., JSON over HTTPS) would require similar mechanisms for authenticating and securing access to configuration data.

Most of the data elements in the configuration model could be considered sensitive from a security standpoint. Unauthorized access or invalid data could cause major disruption.

5. IANA Considerations

An appropriate namespace URI will be registered in the IETF XML Registry" [RFC3688]. The BGP YANG modules will be registered in the "YANG Module Names" registry [RFC6020].

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:

- o `ietf-bgp-common` - defines the groupings that are common across more than one context (where contexts are neighbor, group, global)
- o `ietf-bgp-common-multiprotocol` - defines the groupings that are common across more than one context, and relate to multiprotocol BGP
- o `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)

- o ietf-bgp-global - groupings with data specific to the global context
- o ietf-bgp-peer-group - groupings with data specific to the peer group context
- o ietf-bgp-neighbor - groupings with data specific to the neighbor context

Additional modules include:

- o ietf-bgp-types - common type and identity definitions for BGP, including BGP policy
- o ietf-bgp-policy - BGP-specific policy data definitions for use with [I-D.ietf-rtgwg-policy-model] (described in more detail Section 2.2)

7. BGP main module and submodule for base items

```
<CODE BEGINS> file "ietf-bgp@2017-10-17.yang"
module ietf-bgp {

    yang-version "1";

    // namespace
    namespace "urn:ietf:params:xml:ns:yang:ietf-bgp";

    prefix "bgp";

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

    // Common: defines the groupings that are common across more than
    //           one context (where contexts are neighbor, group, global)
    include ietf-bgp-common;
    // Multiprotocol: defines the groupings that are common across more
    //           than one context, and relate to Multiprotocol
    include ietf-bgp-common-multiprotocol;
    // Structure: defines groupings that are shared but are solely used
    //           for structural reasons.
    include ietf-bgp-common-structure;
    // Include peer-group/neighbor/global - these define the groupings
```

```
// that are specific to one context
include ietf-bgp-neighbor;
include ietf-bgp-global;
include ietf-bgp-peer-group;

// meta
organization
  "IETF IDR Working Group";

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

  Editor:    Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors:   Keyur Patel,
             Mahesh Jethanandani,
             Susan Hares";

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

  This model supports the following BGP configuration level
  hierarchy:

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network ";
```

```
    }

    /*
     * Groupings
     */
    container bgp {
        description
            "Top-level configuration for the BGP router";

        container global {
            description
                "Global configuration for the BGP router";
            uses bgp-global-base;
            uses rpol:apply-policy-group;
        }

        container neighbors {
            description
                "Configuration for BGP neighbors";
            uses bgp-neighbor-list;
        }

        container peer-groups {
            description
                "Configuration for BGP peer-groups";
            uses bgp-peer-group-list;
        }
    }
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-common@2017-10-17.yang"
submodule ietf-bgp-common {
    belongs-to ietf-bgp {
        prefix "bgp";
    }

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

  Editor:    Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors:   Keyur Patel,
             Mahesh Jethanandani,
             Susan Hares";

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

grouping bgp-common-neighbor-group-timers-config {
  description
    "Config parameters related to timers associated with the BGP
    peer";

  leaf connect-retry {
    type decimal64 {
      fraction-digits 2;
    }
    default 30;
    description
      "Time interval in seconds between attempts to establish a
      session with the peer.";
  }

  leaf hold-time {
    type decimal64 {
      fraction-digits 2;
    }
    default 90;
    description
      "Time interval in seconds that a BGP session will be
      considered active in the absence of keepalive or other
      messages from the peer. The hold-time is typically set to
      3x the keepalive-interval.";
    reference

```

```
    "RFC 4271 - A Border Gateway Protocol 4, Sec. 10";
  }

  leaf keepalive-interval {
    type decimal64 {
      fraction-digits 2;
    }
    default 30;
    description
      "Time interval in seconds between transmission of keepalive
      messages to the neighbor. Typically set to 1/3 the
      hold-time.";
  }

  leaf minimum-advertisement-interval {
    type decimal64 {
      fraction-digits 2;
    }
    default 30;
    description
      "Minimum time which must elapse between subsequent UPDATE
      messages relating to a common set of NLRI being transmitted
      to a peer. This timer is referred to as
      MinRouteAdvertisementIntervalTimer by RFC 4721 and serves to
      reduce the number of UPDATE messages transmitted when a
      particular set of NLRI exhibit instability.";
    reference
      "RFC 4271 - A Border Gateway Protocol 4, Sec 9.2.1.1";
  }
}

grouping bgp-common-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 peer-type {
    type bgp-types:peer-type;
    description
        "Explicitly designate the peer or peer group as internal
        (iBGP) or external (eBGP).";
}

leaf auth-password {
    type string;
    description
        "Configures an MD5 authentication password for use with
        neighboring devices.";
}

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

leaf route-flap-damping {
    type boolean;
    default false;
    description
        "Enable route flap damping.";
}

leaf send-community {
    type bgp-types:community-type;
    default "NONE";
    description
        "Specify which types of community should be sent to the
        neighbor or group. The default is to not send the community
        attribute";
}

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



```
    }  
  }  
  
  grouping bgp-common-neighbor-group-transport-config {  
    description  
      "Configuration parameters relating to the transport protocol  
      used by the BGP session to the peer";  
  
    leaf tcp-mss {  
      type uint16;  
      description  
        "Sets the max segment size for BGP TCP sessions.";  
    }  
  
    leaf mtu-discovery {  
      type boolean;  
      default false;  
      description  
        "Turns path mtu discovery for BGP TCP sessions on (true) or  
        off (false)";  
    }  
  
    leaf passive-mode {  
      type boolean;  
      default false;  
      description  
        "Wait for peers to issue requests to open a BGP session,  
        rather than initiating sessions from the local router.";  
    }  
  
    leaf local-address {  
      type union {  
        type inet:ip-address;  
        type string;  
      }  
      //TODO: the string should be converted to a leafref type  
      //to point to an interface when YANG 1.1 is available with  
      //leafrefs in union types.  
      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.";  
    }  
  }  
  
  grouping bgp-common-neighbor-group-error-handling-config {  
    description
```

```
    "Configuration parameters relating to enhanced error handling
    behaviours for BGP";

    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 "draft-ietf-idr-error-handling-16";
    }
}

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

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

    leaf restart-time {
        type uint16 {
            range 0..4096;
        }
        description
            "Estimated time (in seconds) for the local BGP speaker to
            restart a session. This value is advertised 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.";
    }

    leaf stale-routes-time {
        type decimal64 {
            fraction-digits 2;
        }
        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";
    }
}
```

```
    }

    leaf helper-only {
      type boolean;
      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.";
    }
  }

  grouping bgp-common-use-multiple-paths-config {
    description
      "Generic configuration options relating to use of multiple
      paths for a referenced AFI-SAFI, group or neighbor";

    leaf enabled {
      type boolean;
      default false;
      description
        "Whether the use of multiple paths for the same NLRI is
        enabled for the neighbor. This value is overridden by any
        more specific configuration value.";
    }
  }

  grouping bgp-common-use-multiple-paths-ebgp-as-options-config {
    description
      "Configuration parameters specific to eBGP multipath applicable
      to all contexts";

    leaf allow-multiple-as {
      type boolean;
      default "false";
      description
        "Allow multipath to use paths from different neighbouring ASes.
        The default is to only consider multiple paths from the same
        neighbouring AS.";
    }
  }

  grouping bgp-common-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";

      uses bgp-common-use-multiple-paths-config;

      container ebgp {
        description
          "Multipath parameters for eBGP";

        leaf allow-multiple-as {
          type boolean;
          default "false";
          description
            "Allow multipath to use paths from different neighbouring
            ASes. The default is to only consider multiple paths
            from the same neighbouring AS.";
        }

        leaf maximum-paths {
          type uint32;
          default 1;
          description
            "Maximum number of parallel paths to consider when using
            BGP multipath. The default is use a single path.";
        }
      }

      container ibgp {
        description
          "Multipath parameters for iBGP";

        leaf maximum-paths {
          type uint32;
          default 1;
          description
            "Maximum number of parallel paths to consider when using
            iBGP multipath. The default is to use a single path";
        }
      }
    }

    grouping bgp-common-route-selection-options {
      description
        "Configuration and state relating to route selection options";
```

```
container route-selection-options {
  description
    "Parameters relating to options for route selection";

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

  leaf ignore-as-path-length {
    type boolean;
    default "false";
    description
      "Ignore the AS path length when selecting the best path.
      The default is to use the AS path length and prefer paths
      with shorter length.";
  }

  leaf external-compare-router-id {
    type boolean;
    default "true";
    description
      "When comparing similar routes received from external BGP
      peers, use the router-id as a criterion to select the
      active path.";
  }

  leaf advertise-inactive-routes {
    type boolean;
    default "false";
    description
      "Advertise inactive routes to external peers. The default
      is to only advertise active routes.";
  }

  leaf enable-aigp {
    type boolean;
    default false;
    description
      "Flag to enable sending / receiving accumulated IGP
      attribute in routing updates";
  }
}
```

```
    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";
    }
}

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

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

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

```
<CODE BEGINS> file "ietf-bgp-common-multiprotocol@2017-10-17.yang"
submodule ietf-bgp-common-multiprotocol {
    belongs-to ietf-bgp {
        prefix "bgp";
    }

    import ietf-bgp-types {
        prefix bgp-types;
    }
    import ietf-routing-policy {
        prefix rpol;
    }

    include ietf-bgp-common;
```

```
// meta
organization
  "IETF IDR Working Group";

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

  Editor:  Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors: Keyur Patel,
           Mahesh Jethanandani,
           Susan Hares";

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

grouping bgp-common-mp-afi-safi-graceful-restart-config {
  description
    "BGP graceful restart parameters that apply on a per-AFI-SAFI
    basis";

  leaf enabled {
    type boolean;
    default false;
    description
      "This leaf indicates whether graceful-restart is enabled for
      this AFI-SAFI";
  }
}

grouping bgp-common-mp-afi-safi-config {
  description
    "Configuration parameters used for all BGP AFI-SAFIs";

  leaf afi-safi-name {
    type identityref {
      base bgp-types:AFI_SAFI_TYPE;
    }
    description "AFI,SAFI";
  }
}
```

```
    }

    leaf enabled {
      type boolean;
      default false;
      description
        "This leaf indicates whether the IPv4 Unicast AFI,SAFI is
        enabled for the neighbour or group";
    }
  }
}

grouping bgp-common-mp-all-afi-safi-list-contents {
  description
    "A common grouping used for contents of the list that is used
    for AFI-SAFI entries";

  // import and export policy included for the afi/safi
  uses rpol:apply-policy-group;

  uses bgp-common-mp-ipv4-unicast-group;
  uses bgp-common-mp-ipv6-unicast-group;
  uses bgp-common-mp-ipv4-labeled-unicast-group;
  uses bgp-common-mp-ipv6-labeled-unicast-group;
  uses bgp-common-mp-l3vpn-ipv4-unicast-group;
  uses bgp-common-mp-l3vpn-ipv6-unicast-group;
  uses bgp-common-mp-l3vpn-ipv4-multicast-group;
  uses bgp-common-mp-l3vpn-ipv6-multicast-group;
  uses bgp-common-mp-l2vpn-vpls-group;
  uses bgp-common-mp-l2vpn-evpn-group;
}

// Groupings relating to each address family
grouping bgp-common-mp-ipv4-unicast-group {
  description
    "Group for IPv4 Unicast configuration options";

  container ipv4-unicast {
    when "../afi-safi-name = 'bgp-types:IPV4_UNICAST'" {
      description
        "Include this container for IPv4 Unicast specific
        configuration";
    }

    description "IPv4 unicast configuration options";

    // include common IPv[46] unicast options
    uses bgp-common-mp-ipv4-ipv6-unicast-common;
  }
}
```



```
    // placeholder for IPv4 unicast specific configuration
  }
}

grouping bgp-common-mp-ipv6-unicast-group {
  description
    "Group for IPv6 Unicast configuration options";

  container ipv6-unicast {
    when "../afi-safi-name = 'bgp-types:IPV6_UNICAST'" {
      description
        "Include this container for IPv6 Unicast specific
        configuration";
    }

    description "IPv6 unicast configuration options";

    // include common IPv[46] unicast options
    uses bgp-common-mp-ipv4-ipv6-unicast-common;

    // placeholder for IPv6 unicast specific configuration
    // options
  }
}

grouping bgp-common-mp-ipv4-labeled-unicast-group {
  description
    "Group for IPv4 Labeled Unicast configuration options";

  container ipv4-labeled-unicast {
    when "../afi-safi-name = 'bgp-types:IPV4_LABELED_UNICAST'" {
      description
        "Include this container for IPv4 Labeled Unicast specific
        configuration";
    }

    description "IPv4 Labeled Unicast configuration options";

    uses bgp-common-mp-all-afi-safi-common;

    // placeholder for IPv4 Labeled Unicast specific config
    // options
  }
}

grouping bgp-common-mp-ipv6-labeled-unicast-group {
  description
    "Group for IPv6 Labeled Unicast configuration options";
```

```
    container ipv6-labeled-unicast {
      when "../afi-safi-name = 'bgp-types:IPV6_LABELED_UNICAST'" {
        description
          "Include this container for IPv6 Labeled Unicast specific
          configuration";
      }

      description "IPv6 Labeled Unicast configuration options";

      uses bgp-common-mp-all-afi-safi-common;

      // placeholder for IPv6 Labeled Unicast specific config
      // options.
    }
  }

  grouping bgp-common-mp-l3vpn-ipv4-unicast-group {
    description
      "Group for IPv4 Unicast L3VPN configuration options";

    container l3vpn-ipv4-unicast {
      when "../afi-safi-name = 'bgp-types: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 bgp-common-mp-l3vpn-ipv4-ipv6-unicast-common;

      // placeholder for IPv4 Unicast L3VPN specific config options.
    }
  }

  grouping bgp-common-mp-l3vpn-ipv6-unicast-group {
    description
      "Group for IPv6 Unicast L3VPN configuration options";

    container l3vpn-ipv6-unicast {
      when "../afi-safi-name = 'bgp-types: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 bgp-common-mp-l3vpn-ipv4-ipv6-unicast-common;

// placeholder for IPv6 Unicast L3VPN specific configuration
// options
}
}

grouping bgp-common-mp-l3vpn-ipv4-multicast-group {
  description
    "Group for IPv4 L3VPN multicast configuration options";

  container l3vpn-ipv4-multicast {
    when "../afi-safi-name = 'bgp-types: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 bgp-common-mp-l3vpn-ipv4-ipv6-multicast-common;

    // placeholder for IPv4 Multicast L3VPN specific configuration
    // options
  }
}

grouping bgp-common-mp-l3vpn-ipv6-multicast-group {
  description
    "Group for IPv6 L3VPN multicast configuration options";

  container l3vpn-ipv6-multicast {
    when "../afi-safi-name = 'bgp-types: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 bgp-common-mp-l3vpn-ipv4-ipv6-multicast-common;

    // placeholder for IPv6 Multicast L3VPN specific configuration
    // options
  }
}
```

```
grouping bgp-common-mp-l2vpn-vpls-group {
  description
    "Group for BGP-signalled VPLS configuration options";

  container l2vpn-vpls {
    when "../afi-safi-name = 'bgp-types:L2VPN_VPLS'" {
      description
        "Include this container for BGP-signalled VPLS specific
        configuration";
    }

    description "BGP-signalled VPLS configuration options";

    // include common L2VPN options
    uses bgp-common-mp-l2vpn-common;

    // placeholder for BGP-signalled VPLS specific configuration
    // options
  }
}

grouping bgp-common-mp-l2vpn-evpn-group {
  description
    "Group for BGP EVPN configuration options";

  container l2vpn-evpn {
    when "../afi-safi-name = 'bgp-types:L2VPN_EVPN'" {
      description
        "Include this container for BGP EVPN specific
        configuration";
    }

    description "BGP EVPN configuration options";

    // include common L2VPN options
    uses bgp-common-mp-l2vpn-common;

    // placeholder for BGP EVPN specific configuration options
  }
}

// Common groupings across multiple AFI,SAFIs
grouping bgp-common-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
       neighbour";
  }
  leaf shutdown-threshold-pct {
    type bgp-types:percentage;
    description
      "Threshold on number of prefixes that can be received from
       a neighbour before generation of warning messages or log
       entries. Expressed as a percentage of max-prefixes";
  }

  leaf restart-timer {
    type decimal64 {
      fraction-digits 2;
    }
    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 bgp-common-mp-ipv4-ipv6-unicast-common {
  description
    "Common configuration that is applicable for IPv4 and IPv6
    unicast";

  // include common afi-safi options.
  uses bgp-common-mp-all-afi-safi-common;

  // configuration options that are specific to IPv[46] unicast
  leaf send-default-route {
    type boolean;
    default "false";
    description
      "If set to true, send the default-route to the neighbour(s)";
  }
}

grouping bgp-common-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 bgp-common-mp-all-afi-safi-common;
    }

    grouping bgp-common-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 bgp-common-mp-all-afi-safi-common;
    }

    grouping bgp-common-mp-l2vpn-common {
        description
            "Common configuration applied across L2VPN address
            families";

        // placeholder -- specific configuration options that are
        // generic across L2VPN address families
        uses bgp-common-mp-all-afi-safi-common;
    }

    // Config groupings for common groups
    grouping bgp-common-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@2017-10-17.yang"
submodule ietf-bgp-common-structure {

    belongs-to ietf-bgp {
        prefix "bgp";
    }

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

```
include ietf-bgp-common;

// meta
organization
  "IETF IDR Working Group";

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

  Editor:     Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors:    Keyur Patel,
              Mahesh Jethanandani,
              Susan Hares";

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

grouping bgp-common-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.";
    }
  }
}

grouping bgp-common-structure-neighbor-group-ebgp-multihop {
```

```
description
  "Structural grouping used to include eBGP multihop
  configuration and state for both BGP neighbors and peer
  groups";

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

  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 bgp-common-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 BGPgroup";

    leaf route-reflector-cluster-id {
      type bgp-types: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.";
    }

    leaf route-reflector-client {
```



```
        type boolean;
        default "false";
        description
            "Configure the neighbor as a route reflector client.";
    }
}

grouping bgp-common-structure-neighbor-group-as-path-options {
    description
        "Structural grouping used to include AS_PATH manipulation
        configuration and state for both BGP neighbors and peer
        groups";

    container as-path-options {
        description
            "AS_PATH manipulation parameters for the BGP neighbor or
            group";
        leaf allow-own-as {
            type uint8;
            default 0;
            description
                "Specify the number of occurrences of the local BGP
                speaker's AS that can occur within the AS_PATH before it
                is rejected.";
        }

        leaf replace-peer-as {
            type boolean;
            default "false";
            description
                "Replace occurrences of the peer's AS in the AS_PATH with
                the local autonomous system number";
        }
    }
}

grouping bgp-common-structure-neighbor-group-add-paths {
    description
        "Structural grouping used to include ADD-PATHS configuration
        and state for both BGP neighbors and peer groups";

    container add-paths {
        description
            "Parameters relating to the advertisement and receipt of
            multiple paths for a single NLRI (add-paths)";

        leaf receive {
```

```

    type boolean;
    default false;
    description
        "Enable ability to receive multiple path advertisements for
         an NLRI from the neighbor or group";
}

leaf send-max {
    type uint8;
    description
        "The maximum number of paths to advertise to neighbors for
         a single NLRI";
}
leaf eligible-prefix-policy {
    type leafref {
        path "/rpol:routing-policy/rpol:policy-definitions/" +
            "rpol:policy-definition/rpol:name";
    }
    description
        "A reference to a routing policy which can be used to
         restrict the prefixes for which add-paths is enabled";
}
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-peer-group@2017-10-17.yang"
submodule ietf-bgp-peer-group {
  belongs-to ietf-bgp {
    prefix "bgp";
  }

  import ietf-routing-policy {
    prefix rpol;
  }

  // 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>

Editor: Mahesh Jethanandani (mjethanandani@gmail.com)
Authors: Keyur Patel,
Mahesh Jethanandani,
Susan Hares";

description

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

revision "2017-10-17" {

description

"Initial Version";

reference

"RFC XXX, BGP Model for Service Provider Network.";

}

grouping bgp-peer-group-config {

description

"Configuration parameters relating to a base BGP peer group that are not also applicable to any other context (e.g., neighbor)";

leaf peer-group-name {

type string;

description

"Name of the BGP peer-group";

}

}

grouping bgp-peer-group-afi-safi-list {

description

"List of address-families associated with the BGP peer-group";

list afi-safi {

key "afi-safi-name";

description

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

uses bgp-common-mp-afi-safi-config;

container graceful-restart {

description

```
        "Parameters relating to BGP graceful-restart";

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

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

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

    uses bgp-peer-group-config;
    uses bgp-common-neighbor-group-config;
    uses bgp-common-state;

    container timers {
        description
            "Timers related to a BGP peer-group";

        uses bgp-common-neighbor-group-timers-config;
    }

    container transport {
        description
            "Transport session parameters for the BGP peer-group";

        uses bgp-common-neighbor-group-transport-config;
    }

    container error-handling {
        description
            "Error handling parameters used for the BGP peer-group";

        uses bgp-common-neighbor-group-error-handling-config;
    }

    container graceful-restart {
        description
            "Parameters relating the graceful restart mechanism for BGP";

        uses bgp-common-graceful-restart-config;
    }

    uses bgp-common-structure-neighbor-group-logging-options;
}
```

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

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

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

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

      uses bgp-peer-group-base;
    }
  }
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-neighbor@2017-10-17.yang"
submodule ietf-bgp-neighbor {
  belongs-to ietf-bgp {
    prefix "bgp";
  }

  import ietf-routing-policy {
    prefix rpol;
  }
  import ietf-bgp-types {
    prefix bgp-types;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
```

```
    prefix yang;
  }

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

  // meta
  organization
    "IETF IDR Working Group";

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

    Editor: Mahesh Jethanandani (mjethanandani@gmail.com)
    Authors: Keyur Patel,
             Mahesh Jethanandani,
             Susan Hares";

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

  revision "2017-10-17" {
    description
      "Initial Version";
    reference
      "RFC XXX, BGP Model for Service Provider Network.";
  }

  grouping bgp-neighbor-use-multiple-paths {
    description
      "Multipath 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";

      uses bgp-common-use-multiple-paths-config;

      container ebgp {
        description
          "Multipath configuration for eBGP";
```

```
        uses bgp-common-use-multiple-paths-ebgp-as-options-config;
    }
}

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

  leaf UPDATE {
    type uint64;
    description
      "Number of BGP UPDATE messages announcing, withdrawing or
      modifying paths exchanged.";
  }

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

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

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

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

    uses bgp-common-mp-afi-safi-config;

    leaf active {
      type boolean;
      config false;
      description
        "This value indicates whether a particular AFI-SAFI has
        been successfully negotiated with the peer. An AFI-SAFI may
        be enabled in the current running configuration, but a
        session restart may be required in order to negotiate the
        new capability.";
    }
  }
}
```

```
container prefixes {
  config false;
  description "Prefix counters for the BGP session";
  leaf received {
    type uint32;
    description
      "The number of prefixes received from the neighbor";
  }

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

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

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

  uses bgp-common-mp-afi-safi-graceful-restart-config;

  leaf received {
    type boolean;
    config false;
    description
      "This leaf indicates whether the neighbor advertised the
      ability to support graceful-restart for this AFI-SAFI";
  }

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

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



```
    }  
  }  
  
  grouping bgp-neighbor-base {  
    description  
      "Parameters related to a BGP neighbor";  
  
    leaf peer-group {  
      type leafref {  
        path "../..../peer-groups/peer-group/peer-group-name";  
      }  
      description  
        "The peer-group with which this neighbor is associated";  
    }  
  
    leaf neighbor-address {  
      type inet:ip-address;  
      description  
        "Address of the BGP peer, either in IPv4 or IPv6";  
    }  
  
    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.";  
    }  
  
    uses bgp-common-neighbor-group-config;  
  
    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
        connction 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";
    }
}
config false;
description
    "Operational state of the BGP peer";
}

leaf last-established {
    // Was oc-types:timeticks64
    type uint64;
    config false;
    description
        "This timestamp indicates the time that the BGP session last
        transitioned in or out of the Established state. The value
        is the timestamp in seconds relative to the Unix Epoch (Jan
        1, 1970 00:00:00 UTC).

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

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

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

container timers {
    description
        "Timers related to a BGP neighbor";

    uses bgp-common-neighbor-group-timers-config;

    leaf negotiated-hold-time {
        type decimal64 {
            fraction-digits 2;
        }
        config false;
        description
            "The negotiated hold-time for the BGP session";
    }
}

container transport {
    description
        "Transport session parameters for the BGP neighbor";

    uses bgp-common-neighbor-group-transport-config;

    leaf local-port {
        type inet:port-number;
        config false;
        description
            "Local TCP port being used for the TCP session supporting
            the BGP session";
    }

    leaf remote-address {
        type inet:ip-address;
        config false;
        description
            "Remote address to which the BGP session has been
            established";
    }

    leaf remote-port {
        type inet:port-number;
        config false;
        description
            "Remote port being used by the peer for the TCP session
            supporting the BGP session";
    }
}
```

```
    }  
  }  
  
  container error-handling {  
    description  
      "Error handling parameters used for the BGP neighbor or  
      group";  
    uses bgp-common-neighbor-group-error-handling-config;  
  
    leaf erroneous-update-messages {  
      type uint32;  
      config false;  
      description  
        "The number of BGP UPDATE messages for which the  
        treat-as-withdraw mechanism has been applied based on  
        erroneous message contents";  
    }  
  }  
  
  container graceful-restart {  
    description  
      "Parameters relating the graceful restart mechanism for BGP";  
  
    uses bgp-common-graceful-restart-config;  
  
    leaf peer-restart-time {  
      type uint16 {  
        range 0..4096;  
      }  
      config false;  
      description  
        "The period of time (advertised by the peer) that the peer  
        expects a restart of a BGP session to take";  
    }  
  
    leaf peer-restarting {  
      type boolean;  
      config false;  
      description  
        "This flag indicates whether the remote neighbor is  
        currently in the process of restarting, and hence received  
        routes are currently stale";  
    }  
  
    leaf local-restarting {  
      type boolean;  
      config false;  
      description
```

```
        "This flag indicates whether the local neighbor is
        currently restarting. The flag is unset after all NLRI
        have been advertised to the peer, and the End-of-RIB (EOR)
        marker has been unset";
    }

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

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

container afi-safis {
    description
    "Per-address-family configuration parameters associated with
    the neighbor";
```

```
        uses bgp-neighbor-afi-safi-list;
    }
}

grouping bgp-neighbor-list {
    description
        "The list of BGP neighbors";

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

        uses bgp-neighbor-base;
    }
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-global@2017-10-17.yang"
submodule ietf-bgp-global {
    belongs-to ietf-bgp {
        prefix "bgp";
    }

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

    // Include common submodule
    include ietf-bgp-common;
    include ietf-bgp-common-multiprotocol;

    // meta
    organization
        "IETF IDR Working Group";

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

        Editor:    Mahesh Jethanandani (mjethanandani@gmail.com)
        Authors:    Keyur Patel,
```

Mahesh Jethanandani,
Susan Hares";

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

grouping bgp-global-config {
  description
    "Global configuration options 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 router-id {
    type yang:dotted-quad;
    description
      "Router id of the router - an unsigned 32-bit integer
      expressed in dotted quad notation.";
    reference
      "RFC4271 - A Border Gateway Protocol 4 (BGP-4),
      Section 4.2";
  }
}

grouping bgp-global-state {
  description
    "Operational state parameters for the BGP neighbor";

  uses bgp-common-state;
}

grouping bgp-global-default-route-distance-config {
  description
    "Configuration options relating to the administrative distance
    (or preference) assigned to routes received from different
```



```
        sources (external, internal, and local).";

    leaf external-route-distance {
        type uint8 {
            range "1..255";
        }
        description
            "Administrative distance for routes learned from external
            BGP (eBGP).";
    }
    leaf internal-route-distance {
        type uint8 {
            range "1..255";
        }
        description
            "Administrative distance for routes learned from internal
            BGP (iBGP).";
    }
}

grouping bgp-global-confederation-config {
    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.";
    }
}

grouping bgp-global-afi-safi-list {
```

```
description
  "List of address-families associated with the BGP instance";

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

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

  uses bgp-common-mp-afi-safi-config;
  uses bgp-common-state;

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

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

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

// Structural groupings
grouping bgp-global-base {
  description
    "Global configuration parameters for the BGP router";

  uses bgp-global-config;
  uses bgp-global-state;

  container default-route-distance {
    description
      "Administrative distance (or preference) assigned to
      routes received from different sources
      (external, internal, and local).";

    uses bgp-global-default-route-distance-config;
  }

  container confederation {
    description
      "Parameters indicating whether the local system acts as part
      of a BGP confederation";
```

```
    uses bgp-global-confederation-config;
  }

  container graceful-restart {
    description
      "Parameters relating the graceful restart mechanism for BGP";
    uses bgp-common-graceful-restart-config;
  }

  uses bgp-common-global-group-use-multiple-paths;
  uses bgp-common-route-selection-options;

  container afi-safis {
    description
      "Address family specific configuration";
    uses bgp-global-afi-safi-list;
  }
}
<CODE ENDS>
```

8. BGP types

```
<CODE BEGINS> file "ietf-bgp-types@2017-10-17.yang"
module ietf-bgp-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-types";

  prefix "bgp-types";

  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>

    Editor:  Mahesh Jethanandani (mjethanandani@gmail.com)
    Authors: Keyur Patel,
             Mahesh Jethanandani,
             Susan Hares";
```

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

revision "2017-10-17" {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

identity BGP_CAPABILITY {
  description "Base identity for a BGP capability";
}

identity MPBGP {
  base BGP_CAPABILITY;
  description
    "Multi-protocol extensions to BGP";
  reference "RFC2858";
}

identity ROUTE_REFRESH {
  base BGP_CAPABILITY;
  description
    "The BGP route-refresh functionality";
  reference "RFC2918";
}

identity ASN32 {
  base BGP_CAPABILITY;
  description
    "4-byte (32-bit) AS number functionality";
  reference "RFC6793";
}

identity GRACEFUL_RESTART {
  base BGP_CAPABILITY;
  description
    "Graceful restart functionality";
  reference "RFC4724";
}

identity ADD_PATHS {
  base BGP_CAPABILITY;
  description
    "BGP add-paths";
}
```

```
    reference "draft-ietf-idr-add-paths";
  }

  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";
  }

  identity IPV6_UNICAST {
    base AFI_SAFI_TYPE;
    description
      "IPv6 unicast (AFI,SAFI = 2,1)";
    reference "RFC4760";
  }

  identity IPV4_LABELED_UNICAST {
    base AFI_SAFI_TYPE;
    description
      "Labeled IPv4 unicast (AFI,SAFI = 1,4)";
    reference "RFC3107";
  }

  identity IPV6_LABELED_UNICAST {
    base AFI_SAFI_TYPE;
    description
      "Labeled IPv6 unicast (AFI,SAFI = 2,4)";
    reference "RFC3107";
  }

  identity L3VPN_IPV4_UNICAST {
    base AFI_SAFI_TYPE;
    description
      "Unicast IPv4 MPLS L3VPN (AFI,SAFI = 1,128)";
    reference "RFC4364";
  }

  identity L3VPN_IPV6_UNICAST {
    base AFI_SAFI_TYPE;
    description
      "Unicast IPv6 MPLS L3VPN (AFI,SAFI = 2,128)";
```

```
    reference "RFC4659";
  }

  identity L3VPN_IPV4_MULTICAST {
    base AFI_SAFI_TYPE;
    description
      "Multicast IPv4 MPLS L3VPN (AFI,SAFI = 1,129)";
    reference "RFC6514";
  }

  identity L3VPN_IPV6_MULTICAST {
    base AFI_SAFI_TYPE;
    description
      "Multicast IPv6 MPLS L3VPN (AFI,SAFI = 2,129)";
    reference "RFC6514";
  }

  identity L2VPN_VPLS {
    base AFI_SAFI_TYPE;
    description
      "BGP-signalled VPLS (AFI,SAFI = 25,65)";
    reference "RFC4761";
  }

  identity L2VPN_EVPN {
    base AFI_SAFI_TYPE;
    description
      "BGP MPLS Based Ethernet VPN (AFI,SAFI = 25,70)";
  }

  identity BGP_WELL_KNOWN_STD_COMMUNITY {
    description
      "Reserved communities within the standard community space
      defined by RFC1997. These communities must fall within the
      range 0x00000000 to 0xFFFFFFFF";
    reference "RFC1997";
  }

  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 0xFFFFFFF01.";
    reference "RFC1997";
  }
```

```
identity NO_ADVERTISE {
  base BGP_WELL_KNOWN_STD_COMMUNITY;
  description
    "All NLRI received carrying this community must not be
    advertised to other BGP peers. This community has a value of
    0xFFFFFFFF02.";
  reference "RFC1997";
}

identity NO_EXPORT_SUBCONFED {
  base BGP_WELL_KNOWN_STD_COMMUNITY;
  description
    "All NLRI received carrying this community must not be
    advertised to external BGP peers - including over confederation
    sub-AS boundaries. This community has a value of 0xFFFFFFFF03.";
  reference "RFC1997";
}

identity NOPEER {
  base BGP_WELL_KNOWN_STD_COMMUNITY;
  description
    "An autonomous system receiving NLRI tagged with this community
    is advised not to readvertise the NLRI to external bi-lateral
    peer autonomous systems. An AS may also filter received NLRI
    from bilateral peer sessions when they are tagged with this
    community value";
  reference "RFC3765";
}

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

typedef bgp-well-known-community-type {
  type identityref {
    base BGP_WELL_KNOWN_STD_COMMUNITY;
  }
}
```

```

    description
        "Type definition for well-known IETF community attribute
        values";
    reference
        "IANA Border Gateway Protocol (BGP) Well Known Communities";
}

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

typedef bgp-ext-community-type {
    // TODO: needs more work to make this more precise given the
    // variability of extended community attribute specifications
    // 8-octet value:
    // <type> 2 octets
    // <value> 6 octets

    type union {
        type string {
            // Type 1: 2-octet global and 4-octet local
            // (AS number) (Integer)
            pattern '(6[0-5][0-5][0-3][0-5]|[1-5][0-9]{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)
        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)
        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
        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
        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 {
    // TODO: needs more work to decide what format these regexps can
    // take.

```

```
//type oc-types:std-regexp;
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 peer-type {
  type enumeration {
    enum INTERNAL {
      description "internal (iBGP) peer";
    }
    enum EXTERNAL {
      description "external (eBGP) peer";
    }
  }
  description
    "labels a peer or peer group as explicitly internal or
    external";
}

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

identity PRIVATE_AS_REMOVE_ALL {
  base REMOVE_PRIVATE_AS_OPTION;
  description
```

```
    "Strip all private autonomous system numbers from the AS_PATH.
    This action is performed regardless of the other content of the
    AS_PATH attribute, and for all instances of private AS numbers
    within that attribute.";
}

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

typedef remove-private-as-option {
  type identityref {
    base REMOVE_PRIVATE_AS_OPTION;
  }
  description
    "set of options for configuring how private AS path numbers
    are removed from advertisements";
}

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

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

typedef community-type {
  type enumeration {
    enum STANDARD {
      description "send only standard communities";
    }
  }
}
```

```
    }
    enum EXTENDED {
        description "send only extended communities";
    }
    enum BOTH {
        description "send both standard and extended communities";
    }
    enum NONE {
        description "do not send any community attribute";
    }
}
description
    "type describing variations of community attributes:
    STANDARD: standard BGP community [rfc1997]
    EXTENDED: extended BGP community [rfc4360]
    BOTH: both standard and extended community";
}
}
<CODE ENDS>
```

9. BGP policy data

```
<CODE BEGINS> file "ietf-bgp-policy@2017-10-17.yang"
module ietf-bgp-policy {
    yang-version "1.1";

    // namespace
    namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-policy";

    prefix "bgp-pol";

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

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

```
}

// meta
organization
  "IETF IDR Working Group";

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

  Editor:     Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors:    Keyur Patel,
              Mahesh Jethanandani,
              Susan Hares";

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

revision "2017-10-17" {
  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]+";
    }
    type enumeration {
      enum IGP {
        description "set the MED value to the IGP cost toward the
        next hop for the route";
      }
    }
  }
  description
    "Type definition for specifying how the BGP MED can
    be set in BGP policy actions. The three choices are to set
    the MED directly, increment/decrement using +/- notation,
    and setting it to the IGP cost (predefined value).";
}

// grouping statements

grouping match-community-top {
  description
    "Top-level grouping for match conditions on communities";
```

```
    container match-community-set {
      description
        "Top-level container for match conditions on communities.
        Match a referenced community-set according to the logic
        defined in the match-set-options leaf";

      leaf community-set {
        type leafref {
          path
            "/rpol:routing-policy/rpol:defined-sets/" +
            "bgp-pol:bgp-defined-sets/bgp-pol:community-sets/" +
            "bgp-pol:community-set/bgp-pol:community-set-name";
        }
        description
          "References a defined community set";
      }

      uses rpol:match-set-options-group;
    }
  }

  grouping match-ext-community-top {
    description
      "Top-level grouping for match conditions on extended
      communities";

    container match-ext-community-set {
      description
        "Match a referenced extended community-set according to the
        logic defined in the match-set-options leaf";

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

      uses rpol:match-set-options-group;
    }
  }

  grouping match-as-path-top {
    description
```

```
    "Top-level grouping for match conditions on AS path set";

    container match-as-path-set {
        description
            "Match a referenced as-path set according to the logic
            defined in the match-set-options leaf";

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

grouping bgp-match-set-conditions {
    description
        "Condition statement definitions for checking membership in a
        defined set";

    uses match-community-top;
    uses match-ext-community-top;
    uses match-as-path-top;
}

grouping community-count-top {
    description
        "Top-level grouping for community count condition";

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

        uses pol-types:attribute-compare-operators;
    }
}

grouping as-path-length-top {
    description
        "Top-level grouping for AS path length condition";

    container as-path-length {
        description
```



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

        uses pol-types:attribute-compare-operators;
    }
}

grouping bgp-conditions-top {
    description
        "Top-level grouping for BGP-specific policy conditions";

    container bgp-conditions {
        description
            "Top-level container ";

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

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

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

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

        leaf local-pref-eq {
            type uint32;
            // TODO: add support for other comparisons if needed
            description

```

```
        "Condition to check if the local pref attribute is equal to
        the specified value";
    }

    leaf route-type {
        // TODO: verify extent of vendor support for this comparison
        type enumeration {
            enum INTERNAL {
                description "route type is internal";
            }
            enum EXTERNAL {
                description "route type is external";
            }
        }
        description
            "Condition to check the route type in the route update";
    }

    uses community-count-top;
    uses as-path-length-top;
    uses bgp-match-set-conditions;
}

grouping community-set-top {
    description
        "Top-level grouping for BGP community sets";

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

        list community-set {
            key "community-set-name";
            description
                "List of defined BGP community sets";

            leaf community-set-name {
                type string;
                mandatory true;
                description
                    "name / label of the community set -- this is used to
                    reference the set in match conditions";
            }

            leaf-list community-member {
                type union {
                    type bgp-types:bgp-std-community-type;
                }
            }
        }
    }
}
```

```
        type bgp-types:bgp-community-regexp-type;
        type bgp-types:bgp-well-known-community-type;
    }
    description
        "members of the community set";
}
}
}

grouping ext-community-set-top {
    description
        "Top-level grouping for extended BGP community sets";

    container ext-community-sets {
        description
            "Enclosing container for list of extended BGP community
            sets";
        list ext-community-set {
            key "ext-community-set-name";
            description
                "List of defined extended BGP community sets";

            leaf ext-community-set-name {
                type string;
                description
                    "name / label of the extended community set -- this is
                    used to reference the set in match conditions";
            }

            leaf-list ext-community-member {
                type union {
                    type rt-types:route-target;
                    type bgp-types:bgp-community-regexp-type;
                }
                description
                    "members of the extended community set";
            }
        }
    }
}

grouping as-path-set-top {
    description
        "Top-level grouping for AS path sets";

    container as-path-sets {
        description
```

```
    "Enclosing container for list of define AS path sets";

    list as-path-set {
        key "as-path-set-name";
        description
            "List of defined AS path sets";

        leaf as-path-set-name {
            type string;
            description
                "name of the AS path set -- this is used to reference the
                 set in match conditions";
        }

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

// augment statements

augment "/rpol:routing-policy/rpol:defined-sets" {
    description "adds BGP defined sets container to routing policy
    model";

    container bgp-defined-sets {
        description
            "BGP-related set definitions for policy match conditions";

        uses community-set-top;
        uses ext-community-set-top;
        uses as-path-set-top;
    }
}

grouping as-path-prepend-top {
    description
        "Top-level grouping for the AS path prepend action";

    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.";
    }
  }
}

grouping set-community-action-common {
  description
    "Common leaves for set-community and set-ext-community
    actions";

  leaf method {
    type enumeration {
      enum INLINE {
        description
          "The extended communities are specified inline as a
          list";
      }
      enum REFERENCE {
        description
          "The extended communities are specified by referencing a
          defined ext-community set";
      }
    }
    description
      "Indicates the method used to specify the extended
      communities for the set-ext-community action";
  }

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

grouping set-community-inline-top {
  description
    "Top-level grouping or inline specification of set-community
    action";
}
```

```
    container inline {
      when "../config/method=INLINE" {
        description
          "Active only when the set-community method is INLINE";
      }
      description
        "Set the community values for the action inline with
        a list.";

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

grouping set-community-reference-top {
  description
    "Top-level grouping for referencing a community-set in the
    set-community action";

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

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

```
grouping set-community-action-top {
  description
    "Top-level grouping for the set-community action";

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

    uses set-community-action-common;
    uses set-community-inline-top;
    uses set-community-reference-top;
  }
}

grouping set-ext-community-inline-top {
  description
    "Top-level grouping or inline specification of
    set-ext-community action";

  container inline {
    when "../config/method=INLINE" {
      description
        "Active only when the set-community method is INLINE";
    }
    description
      "Set the extended community values for the action inline with
      a list.";

    leaf-list communities {
      type union {
        type rt-types:route-target;
        type bgp-types:bgp-well-known-community-type;
      }
      description
        "Set the extended community values for the update inline
        with a list.";
    }
  }
}

grouping set-ext-community-reference-top {
  description
    "Top-level grouping for referencing an extended community-set
    in the set-community action";
```

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

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

  grouping set-ext-community-action-top {
    description
      "Top-level grouping for the set-ext-community action";

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

      uses set-community-action-common;
      uses set-ext-community-inline-top;
      uses set-ext-community-reference-top;
    }
  }

  grouping bgp-actions-top {
    description
      "Top-level grouping for BGP-specific actions";

    container bgp-actions {
      description
        "Top-level container for BGP-specific actions";
    }
  }
}
```



```
    leaf set-route-origin {
      type bgp-types:bgp-origin-attr-type;
      description
        "set the origin attribute to the specified value";
    }

    leaf set-local-pref {
      type uint32;
      description
        "set the local pref attribute on the route update";
    }

    leaf set-next-hop {
      type bgp-next-hop-type;
      description
        "set the next-hop attribute in the route update";
    }

    leaf set-med {
      type bgp-set-med-type;
      description
        "set the med metric attribute in the route update";
    }
    uses as-path-prepend-top;
    uses set-community-action-top;
    uses set-ext-community-action-top;
  }
}

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

  uses bgp-conditions-top;
}

augment "/rpol:routing-policy/rpol:policy-definitions/" +
  "rpol:policy-definition/rpol:statements/rpol:statement/" +
  "rpol:actions" {
  description "BGP policy actions added to routing policy
    module";

  uses bgp-actions-top;
}

// rpc statements
```

```
// notification statements
}  
<CODE ENDS>
```

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>>.
- [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>>.
- [RFC3065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", RFC 3065, DOI 10.17487/RFC3065, February 2001, <<https://www.rfc-editor.org/info/rfc3065>>.
- [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>>.
- [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>>.
- [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>>.

- [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>>.
- [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>>.

10.2. Informative references

- [I-D.ietf-netmod-revised-datastores]
Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-06 (work in progress), October 2017.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.
- [I-D.ietf-rtgwg-ni-model]
Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Network Instances", draft-ietf-rtgwg-ni-model-04 (work in progress), September 2017.
- [I-D.ietf-rtgwg-policy-model]
Shaikh, A., Shakir, R., D'Souza, K., and C. Chase, "Routing Policy Configuration Model for Service Provider Networks", draft-ietf-rtgwg-policy-model-01 (work in progress), April 2016.
- [I-D.rtgyangdt-rtgwg-device-model]
Lindem, A., Berger, L., Bogdanovic, D., and C. Hopps, "Network Device YANG Organizational Models", draft-rtgyangdt-rtgwg-device-model-05 (work in progress), August 2016.

Appendix A. 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.

Appendix B. Change summary

B.1. Changes between revisions -01 and -02

- o Refactored BGP model such that it is comprised of multiple sub-modules rather than independent modules.
- o Remove the need for self-augmentation of the BGP model to allow the ability to import the model in wider structures more easily.
- o Added new operational state values for BGP session established transitions and last-established timestamp. Also deprecated uptime operational state leaf.
- o Added ability to select eligible paths for add-paths based on a policy.

B.2. Changes between revisions -00 and -01

- o Updated module namespaces to reflect IETF standard namespace.
- o Updated module filenames with ietf- prefix per RFC 6087 guidelines.

Authors' Addresses

Keyur Patel (editor)
Arrcus
CA
USA

Email: keyur@arrcus.com

Mahesh Jethanandani (editor)
CA
USA

Email: mjethanandani@gmail.com

Susan Hares (editor)
Hickory Hill Consulting
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com