

Interdomain Routing
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2020

M. Jethanandani
Kloud Services
K. Patel
Arrcus
S. Hares
Huawei
J. Haas
Juniper Networks
June 28, 2020

BGP YANG Model for Service Provider Networks
draft-ietf-idr-bgp-model-09

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

This Internet-Draft will expire on December 30, 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Goals and approach	3
1.2.	Note to RFC Editor	4
1.3.	Terminology	5
1.4.	Abbreviations	5
2.	Model overview	5
2.1.	BGP protocol configuration	6
2.2.	Policy configuration overview	9
2.3.	BGP RIB overview	10
2.3.1.	Local Routing	12
2.3.2.	Pre updates per-neighbor	12
2.3.3.	Post updates per-neighbor	12
2.3.4.	Pre route advertisements per-neighbor	12
2.3.5.	Post route advertisements per-neighbor	12
3.	Relation to other YANG data models	12
4.	Security Considerations	13
5.	IANA Considerations	13
5.1.	URI Registration	14
5.2.	YANG Module Name Registration	14
6.	YANG modules	14
7.	Structure of the YANG modules	15
7.1.	Main module and submodules for base items	16
7.2.	BGP types	64
7.3.	BGP policy data	76
7.4.	RIB modules	86
8.	Contributors	120
9.	Acknowledgements	120
10.	References	121
10.1.	Normative references	121
10.2.	Informative references	124
Appendix A.	Examples	125
A.1.	Creating BGP Instance	125
A.2.	Neighbor Address Family Configuration	126
A.3.	IPv6 Neighbor Configuration	128
A.4.	VRF Configuration	129
Appendix B.	How to add a new AFI and Augment a Module	131
Appendix C.	How to deviate a module	133
	Authors' Addresses	133

1. Introduction

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

1.1. Goals and approach

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

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

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

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

- 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.
- o RIB contents.

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

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

1.2. Note to RFC Editor

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

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

2020-06-28 with the actual date of the publication of this document.

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

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

1.3. Terminology

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

1.4. Abbreviations

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

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 Multiprotocol Extensions for BGP-4 [[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 A YANG Data Model for Routing Policy Management [[I-D.ietf-rtgwg-policy-model](#)] that act on routes sent (received) to (from) peers or other routing protocols and BGP-specific policy features.
- o operational state -- variables used for monitoring and management of BGP operations.

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

[2.1.](#) BGP protocol configuration

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


```
module: ietf-bgp
```

```
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
```

```
col:
  +--rw bgp
    +--rw global!
      | +--rw as inet:as-number
      | +--rw identifier? yang:dotted-quad
      | +--rw distance
      | +--rw confederation
      | +--rw graceful-restart {bt:graceful-restart}?
      | +--rw use-multiple-paths
      | +--rw route-selection-options
      | +--rw afi-safis
      | +--rw apply-policy
      | +--ro total-paths? uint32
      | +--ro total-prefixes? uint32
    +--rw neighbors
      | +--rw neighbor* [remote-address]
      | +---n established
      | +---n backward-transition
      | +---x clear {bt:clear-neighbors}?
    +--rw peer-groups
      | +--rw peer-group* [peer-group-name]
    +--rw interfaces
      | +--rw interface* [name]
    +--ro rib
      +--ro attr-sets
      +--ro communities
      +--ro ext-communities
      +--ro afi-safis
```

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

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

is for sub-address family configuration where some items are only applicable for a given AFI-SAFI combination.

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

Address-family configuration is made available in multiple points within the model - primarily within the global container, where instance-wide configuration can be set (for example, global protocol parameters, the BGP best path route selection options, or global policies relating to the address-family); and on a per-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          identityref
        |
        +--rw ipv4-unicast
        |   ...
        +--rw ipv6-unicast
        |   ...
        +--rw ipv4-labeled-unicast
        |   ...
        +--rw ipv6-labeled-unicast
        |   ...
        +--rw l3vpn-ipv4-unicast
        |   ...
        +--rw l3vpn-ipv6-unicast
        |   ...
        +--rw l3vpn-ipv4-multicast
        |   ...
        +--rw l3vpn-ipv6-multicast
        |   ...
        +--rw l2vpn-vpls
        |   ...
        +--rw l2vpn-evpn
        |   ...
```

2.2. Policy configuration overview

The BGP policy configuration model augments the generic YANG routing policy model described in A YANG Data Model for Routing Policy Management [[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 a specific neighbor or group.

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

2.3. BGP RIB overview

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

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


```

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

```


2.3.1. Local Routing

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

2.3.2. Pre updates per-neighbor

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

2.3.3. Post updates per-neighbor

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

2.3.4. Pre route advertisements per-neighbor

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

2.3.5. Post route advertisements per-neighbor

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

3. Relation to other YANG data models

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

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [[RFC6241](#)] or RESTCONF [[RFC8040](#)]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [[RFC6242](#)]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [[RFC8446](#)]. The NETCONF Access Control Model (NACM) [[RFC8341](#)] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

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

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

BGP OPSEC [[RFC7454](#)] describes several policies that can be used to secure a BGP. In particular, it recommends securing the underlying TCP session and to use Generalized TTL Security Mechanism (GTSM) [[RFC5082](#)] capability to make it harder to spoof a BGP session. This module allows implementations that want to support the capability to configure a TTL value, under a feature flag. It also defines a container 'secure-session' that can be augmented with TCP-Authentication Option (TCP-AO) [[RFC5925](#)], or other methods to secure a BGP session, and will be developed in a future version of this draft.

5. IANA Considerations

This document registers three URIs and three YANG modules.

5.1. URI Registration

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

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

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

5.2. YANG Module Name Registration

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

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

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

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

6. YANG modules

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

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

- 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
- o ietf-bgp-rib - grouping for representing BGP RIB.

Additionally, 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. Structure of the YANG modules

The YANG model can be subdivided between the main module for base items, types, policy data, and the RIB module. It references BGP Communities Attribute [[RFC1997](#)], Route Refresh Capability for BGP-4 [[RFC2918](#)], , NOPEER Community for BGP [[RFC3765](#)], BGP/MPLS IP Virtual Private Networks (VPNs) [[RFC4364](#)], BGP/MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN [[RFC4659](#)], Graceful Restart Mechanism for BGP [[RFC4724](#)], Multiprotocol Extensions for BGP-4 [[RFC4760](#)], Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling [[RFC4761](#)], Autonomous System Configuration for BGP [[RFC5065](#)], The Generalized TTL Security Mechanism (GTSM) [[RFC5082](#)], Bidirectional Forward Detection (BFD) [[RFC5880](#)], Bidirectional Forward Detection for IPv4 and IPv6 (Single Hop) [[RFC5881](#)], Bidirectional Forwarding Detection (BFD) for Multihop Paths [[RFC5883](#)], The TCP Authentication Option [[RFC5925](#)], BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs [[RFC6514](#)], BGP Support for Four-Octet Autonomous System (AS) Number Space [[RFC6793](#)], Advertisement of Multiple Paths in BGP [[RFC7911](#)], YANG Key Chain [[RFC8177](#)], and Carrying Label Information in BGP-4 [[RFC8277](#)].

7.1. Main module and submodules for base items

```
<CODE BEGINS> file "ietf-bgp@2020-06-28.yang"
module ietf-bgp {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp";
  prefix bgp;

  /*
   * Import and Include
   */

  import ietf-routing {
    prefix rt;
    reference
      "RFC 8349, A YANG Data Model for Routing Management
      (NMDA Version)";
  }
  import ietf-routing-policy {
    prefix rpol;
    reference
      "RFC ZZZZ, A YANG Data Model for Routing Policy Management";
  }
  import ietf-interfaces {
    prefix if;
    reference
      "RFC 8343, A YANG Data Model for Interface Management.";
  }
  import ietf-bgp-types {
    prefix bt;
    reference
      "RFC XXXX, BGP YANG Model for Service Provider Network.";
  }
  import ietf-bfd-types {
    prefix bfd;
    reference
      "RFC BBBB, YANG Data Model for Bidirectional Forward Detection.";
  }
  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types.";
  }
  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types.";
  }
}
```



```
import ietf-key-chain {
  prefix kc;
  reference
    "RFC 8177: YANG Key Chain.";
}
import ietf-tcp {
  prefix tcp;
  reference
    "I-D.scharf-tcpm-yang-tcp: Transmission Control Protocol (TCP)
    YANG Model.";
}
include ietf-bgp-common {
  revision-date 2020-06-28;
}
include ietf-bgp-common-multiprotocol {
  revision-date 2020-06-28;
}
include ietf-bgp-common-structure {
  revision-date 2020-06-28;
}
include ietf-bgp-neighbor {
  revision-date 2020-06-28;
}
include ietf-bgp-peer-group {
  revision-date 2020-06-28;
}
include ietf-bgp-rib-types {
  revision-date 2020-06-28;
}
include ietf-bgp-rib {
  revision-date 2020-06-28;
}
include ietf-bgp-rib-ext {
  revision-date 2020-06-28;
}
include ietf-bgp-rib-attributes {
  revision-date 2020-06-28;
}
include ietf-bgp-rib-table-attributes {
  revision-date 2020-06-28;
}
include ietf-bgp-rib-tables {
  revision-date 2020-06-28;
}

organization
  "IETF IDR Working Group";
contact
```


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

WG List: <idr@ietf.org>

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

description

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

This model supports the following BGP configuration level hierarchy:

BGP

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

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

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

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

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',

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

```
revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network ";
}

/*
 * Identity
 */

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

/*
 * Containers
 */

augment "/rt:routing/rt:control-plane-protocols/"
  + "rt:control-plane-protocol" {
  when "derived-from-or-self(rt:type, 'bgp')" {
    description
      "This augmentation is valid for a routing protocol
      instance of BGP.";
  }
  description
    "BGP protocol augmentation of ietf-routing module
    control-plane-protocol.";
  container bgp {
    description
      "Top-level configuration for the BGP router";
    container global {
      presence "Enables global configuration of BGP";
      description
        "Global configuration for the BGP router";
      leaf as {
        type inet:as-number;
        mandatory true;
        description
          "Local autonomous system number of the router. Uses
          the 32-bit as-number type from the model in RFC 6991.";
      }
    }
  }
}
```



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



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

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



```
list neighbor {
  key "remote-address";

  description
    "List of BGP neighbors configured on the local system,
    uniquely identified by remote IPv[46] address";

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

  leaf local-port {
    type inet:port-number {
      range "0..65535";
    }
    config false;
    description
      "The local port for the TCP connection between
      the BGP peers.";
  }

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

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

  leaf identifier {
    type yang:dotted-quad;
    config false;
```



```
    description
      "The BGP Identifier of this entry's BGP peer.
      This entry MUST be 0.0.0.0 unless the
      sessionstate is in the openconfirm or the
      established state.";
    reference
      "RFC 4271, Section 4.2, 'BGP Identifier'.";
  }

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

  leaf remote-port {
    type inet:port-number {
      range "0..65535";
    }
    config false;
    description
      "The remote port for the TCP connection
      between the BGP peers. Note that the
      objects local-addr, local-port, remote-addr, and
      reemote-port provide the appropriate
      reference to the standard MIB TCP
      connection table.";
  }

  leaf enabled {
    type boolean;
    default "true";
    description
      "Whether the BGP peer is enabled. In cases where the
      enabled leaf is set to false, the local system should
      not initiate connections to the neighbor, and should
      not respond to TCP connections attempts from the
      neighbor. If the state of the BGP session is
      ESTABLISHED at the time that this leaf is set to false,
      the BGP session should be ceased.

      A transition from 'false' to 'true' will cause
      the BGP Manual Start Event to be generated.
      A transition from 'true' to 'false' will cause
      the BGP Manual Stop Event to be generated.
      This parameter can be used to restart BGP peer
      connections. Care should be used in providing
      write access to this object without adequate
```



```
        authentication.";
    reference
        "RFC 4271, Section 8.1.2.";
}

leaf secure-session-enable {
    type boolean;
    default "false";
    description
        "Does this session need to be secured?";
}

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

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

    case md5 {
        uses tcp:md5;
        leaf md5-keychain {
            type kc:key-chain-ref;
            description
                "Reference to the key chain that will be used by
                this model. Applicable for TCP-AO and TCP-MD5
                only";
            reference
                "RFC 8177: YANG Key Chain.";
        }
    }
}
```



```
    }
    description
      "Uses TCP-MD5 to secure the session. Parameters for
       those are defined as a grouping in the TCP YANG
       model.";
    reference
      "RFC 5925: The TCP Authentication Option.";
  }

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

leaf ttl-security {
  if-feature "bt:ttl-security";
  type uint8;
  default "255";
  description
    "BGP Time To Live (TTL) security check.";
  reference
    "RFC 5082: The Generalized TTL Security Mechanism
     (GTSM),
     RFC 7454: BGP Operations and Security.";
}
uses neighbor-group-config;
uses route-selection-options;
leaf session-state {
  type enumeration {
    enum idle {
      description
        "Neighbor is down, and in the Idle state of the FSM";
    }
    enum connect {
      description
        "Neighbor is down, and the session is waiting for the
```



```
        underlying transport session to be established";
    }
    enum active {
        description
            "Neighbor is down, and the local system is awaiting a
            connection from the remote peer";
    }
    enum opensent {
        description
            "Neighbor is in the process of being established.  The
            local system has sent an OPEN message";
    }
    enum openconfirm {
        description
            "Neighbor is in the process of being established.
            The local system is awaiting a NOTIFICATION or
            KEEPALIVE message";
    }
    enum established {
        description
            "Neighbor is up - the BGP session with the peer is
            established";
    }
}
// notification does not like a non-config statement.
// config false;
description
    "The BGP peer connection state.";
reference
    "RFC 4271, Section 8.1.2.";
}
leaf last-established {
    type uint64;
    config false;
    description
        "This timestamp indicates the time that the BGP session
        last transitioned in or out of the Established state.
        The value is the timestamp in seconds relative to the
        Unix Epoch (Jan 1, 1970 00:00:00 UTC).

        The BGP session uptime can be computed by clients as
        the difference between this value and the current time
        in UTC (assuming the session is in the ESTABLISHED
        state, per the session-state leaf).";
}
leaf-list supported-capabilities {
    type identityref {
        base bt:bgp-capability;
```



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



```
}
container transport {
  description
    "Transport session parameters for the BGP neighbor";
  uses neighbor-group-transport-config;
}
leaf treat-as-withdraw {
  type boolean;
  default "false";
  description
    "Specify whether erroneous UPDATE messages for which
    the NLRI can be extracted are treated as though the
    NLRI is withdrawn - avoiding session reset";
  reference
    "RFC 7606: Revised Error Handling for BGP UPDATE
    Messages.";
}
leaf erroneous-update-messages {
  type uint32;
  config false;
  description
    "The number of BGP UPDATE messages for which the
    treat-as-withdraw mechanism has been applied based on
    erroneous message contents";
}
container graceful-restart {
  if-feature "bt:graceful-restart";
  description
    "Parameters relating the graceful restart mechanism for
    BGP";
  uses graceful-restart-config;
  leaf peer-restart-time {
    type uint16 {
      range "0..4096";
    }
    config false;
    description
      "The period of time (advertised by the peer) that the
      peer expects a restart of a BGP session to take";
  }
  leaf peer-restarting {
    type boolean;
    config false;
    description
      "This flag indicates whether the remote neighbor is
      currently in the process of restarting, and hence
      received routes are currently stale";
  }
}
```



```
leaf local-restarting {
  type boolean;
  config false;
  description
    "This flag indicates whether the local neighbor is
    currently restarting. The flag is unset after all NLRI
    have been advertised to the peer, and the End-of-RIB
    (EOR) marker has been unset";
}
leaf mode {
  type enumeration {
    enum helper-only {
      description
        "The local router is operating in helper-only
        mode, and hence will not retain forwarding state
        during a local session restart, but will do so
        during a restart of the remote peer";
    }
    enum bilateral {
      description
        "The local router is operating in both helper
        mode, and hence retains forwarding state during
        a remote restart, and also maintains forwarding
        state during local session restart";
    }
    enum remote-helper {
      description
        "The local system is able to retain routes during
        restart but the remote system is only able to
        act as a helper";
    }
  }
  config false;
  description
    "This leaf indicates the mode of operation of BGP
    graceful restart with the peer";
}
}
uses structure-neighbor-group-logging-options;
uses structure-neighbor-group-ebgp-multihop;
uses structure-neighbor-group-route-reflector;
uses structure-neighbor-group-as-path-options;
uses structure-neighbor-group-add-paths;
uses bgp-neighbor-use-multiple-paths;
uses rpol:apply-policy-group;
container afi-safis {
  description
    "Per-address-family configuration parameters associated
```



```
        with the neighbor";
    uses bgp-neighbor-afi-safi-list;
}
container statistics {
    description
        "Statistics per neighbor.";

    leaf established-transitions {
        type yang:counter64;
        config false;
        description
            "Number of transitions to the Established state for
            the neighbor session. This value is analogous to the
            bgpPeerFsmEstablishedTransitions object from the
            standard BGP-4 MIB";
        reference
            "RFC 4273 - Definitions of Managed Objects for BGP-4";
    }
    leaf fsm-established-transitions {
        type yang:counter32;
        config false;
        description
            "The total number of times the BGP FSM
            transitioned into the established state
            for this peer.";
        reference
            "RFC 4271, Section 8.";
    }
}
container messages {
    config false;
    description
        "Counters for BGP messages sent and received from the
        neighbor";
    leaf in-total-messages {
        type yang:counter32;
        config false;
        description
            "The total number of messages received
            from the remote peer on this connection.";
        reference
            "RFC 4271, Section 4.";
    }
    leaf out-total-messages {
        type yang:counter32;
        config false;
        description
            "The total number of messages transmitted to
            the remote peer on this connection.";
```



```
        reference
          "RFC 4271, Section 4.";
      }
      leaf in-update-elapsed-time {
        type yang:gauge32;
        units "seconds";
        config false;
        description
          "Elapsed time (in seconds) since the last BGP
           UPDATE message was received from the peer.
           Each time in-updates is incremented,
           the value of this object is set to zero (0).";
        reference
          "RFC 4271, Section 4.3.
           RFC 4271, Section 8.2.2, Established state.";
      }
      container sent {
        description
          "Counters relating to BGP messages sent to the
           neighbor";
        uses bgp-neighbor-counters-message-types-state;
      }
      container received {
        description
          "Counters for BGP messages received from the
           neighbor";
        uses bgp-neighbor-counters-message-types-state;
      }
    }
    container queues {
      config false;
      description
        "Counters related to queued messages associated with
         the BGP neighbor";
      leaf input {
        type uint32;
        description
          "The number of messages received from the peer
           currently queued";
      }
      leaf output {
        type uint32;
        description
          "The number of messages queued to be sent to the
           peer";
      }
    }
  }
  action clear {
```



```
    if-feature "bt:clear-statistics";
    description
      "Clear statistics action command.

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

    input {
      leaf clear-at {
        type yang:date-and-time;
        description
          "Time when the clear action needs to be
          executed.";
      }
    }
    output {
      leaf clear-finished-at {
        type yang:date-and-time;
        description
          "Time when the clear action command completed.";
      }
    }
  }
}
notification established {
  leaf remote-address {
    type leafref {
      path "../../neighbor/remote-address";
    }
    description
      "IP address of the neighbor that went into established
      state.";
  }
  leaf last-error {
    type leafref {
      path "../../neighbor/last-error";
    }
    description
      "The last error code and subcode seen by this
      peer on this connection.  If no error has
      occurred, this field is zero.  Otherwise, the
      first byte of this two byte OCTET STRING
      contains the error code, and the second byte
      contains the subcode.";
    reference
      "RFC 4271, Section 4.5";
  }
}
```



```
leaf session-state {
  type leafref {
    path "../../neighbor/session-state";
  }
  description
    "The BGP peer connection state.";
  reference
    "RFC 4271, Section 8.2.2.";
}
description
  "The established event is generated
  when the BGP FSM enters the established state.";
}

notification backward-transition {
  leaf remote-addr {
    type leafref {
      path "../../neighbor/remote-address";
    }
    description
      "IP address of the neighbor that went away from
      established state.";
  }
  leaf last-error {
    type leafref {
      path "../../neighbor/last-error";
    }
    description
      "The last error code and subcode seen by this
      peer on this connection.  If no error has
      occurred, this field is zero.  Otherwise, the
      first byte of this two byte OCTET STRING
      contains the error code, and the second byte
      contains the subcode.";
    reference
      "RFC 4271, Section 4.5.";
  }
  leaf session-state {
    type leafref {
      path "../../neighbor/session-state";
    }
    description
      "The BGP peer connection state.";
    reference
      "RFC 4271, Section 8.2.2.";
  }
  description
    "The backward-transition event is
```



```
        generated when the BGP FSM moves from a higher
        numbered state to a lower numbered state.";
    }
    action clear {
        if-feature "bt:clear-neighbors";
        description
            "Clear neighbors action.";

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

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

container interfaces {
    list interface {
        key "name";
        leaf name {
            type if:interface-ref;
            description
                "Reference to the interface within the routing instance.";
        }
    }
    container bfd {
        if-feature "bt:bfd";
        uses bfd:client-cfg-parms;
        description
            "BFD client configuration.";
        reference
            "RFC BBBB - YANG Data Model for Bidirectional Forwarding
            Detection.";
    }
    description
```



```
        "List of interfaces within the routing instance.";
    }
    description
        "Interface specific parameters.";
    }
    uses rib;
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-common@2020-06-28.yang"
submodule ietf-bgp-common {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }

    import ietf-bgp-types {
        prefix bt;
        reference
            "RFC XXXX: BGP Model for Service Provider Network.";
    }
    import ietf-inet-types {
        prefix inet;
        reference
            "RFC 6991: Common YANG Data Types.";
    }
    import ietf-yang-types {
        prefix yang;
        reference
            "RFC 6991: Common YANG Data Types.";
    }

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

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arccus.com),
                Susan Hares (shares at ndzh.com,
                Jeffrey Haas (jhaas at pfr.org).";
    description
        "This sub-module contains common groupings that are common across
```


multiple contexts within the BGP module. That is to say that they may be application to a subset of global, peer-group or neighbor contexts.

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

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

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

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

```
revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP Model for Service Provider Network.";
}

grouping neighbor-group-timers-config {
  description
    "Config parameters related to timers associated with the BGP
    peer";
  leaf connect-retry-interval {
    type uint16 {
      range "1..max";
    }
    units "seconds";
    default "120";
    description
      "Time interval (in seconds) for the ConnectRetryTimer. The
      suggested value for this timer is 120 seconds.";
    reference
      "RFC 4271, Section 8.2.2. This is the value used
      to initialize the 'ConnectRetryTimer'.";
  }
}
```



```
leaf hold-time {
  type uint16 {
    range "0 | 3..65535";
  }
  units "seconds";
  default "90";
  description
    "Time interval (in seconds) for the HoldTimer established
    with the peer. When read as operational data (ro), the
    value of this object is calculated by this BGP speaker,
    using the smaller of the values in hold-time that was
    configured (rw) in the running datastore and the Hold Time
    received in the OPEN message.

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

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

    If the configured value of hold-time object was
    a value of (0), then when read this object MUST have a
    value of (0) also.";
  reference
    "RFC 4271, Section 4.2.
    RFC 4271, Section 10.";
}
leaf keepalive {
  type uint16 {
    range "0..21845";
  }
  units "seconds";
  default "30";
  description
    "When used as a configuration (rw) value, this Time interval
    (in seconds) for the KeepAlive timer configured for this BGP
    speaker with this peer. The value of this object will only
    determine the KEEPALIVE messages' frequency relative to
    the value specified in configured value for hold-time.

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

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



```
    of this object is calculated by this BGP speaker such that,
    when compared with hold-time, it has the same proportion
    that keepalive has, compared with hold-time. A
    reasonable maximum value for this timer would be one third
    of that of hold-time.";
  reference
    "RFC 4271, Section 4.4.
     RFC 4271, Section 10.";
}
leaf min-as-origination-interval {
  type uint16 {
    range "0..max";
  }
  units "seconds";
  default "15";
  description
    "Time interval (in seconds) for the MinASOriginationInterval
    timer. The suggested value for this timer is 15 seconds.";
  reference
    "RFC 4271, Section 9.2.1.2.
     RFC 4271, Section 10.";
}
leaf min-route-advertisement-interval {
  type uint16 {
    range "0..max";
  }
  units "seconds";
  description
    "Time interval (in seconds) for the
    MinRouteAdvertisementInterval timer.
    The suggested value for this timer is 30
    seconds for EBGp connections and 5
    seconds for IBGP connections.";
  reference
    "RFC 4271, Section 9.2.1.1.
     RFC 4271, Section 10.";
}
}

grouping neighbor-group-config {
  description
    "Neighbor level configuration items.";
  leaf remote-as {
    type inet:as-number;
    description
      "The remote autonomous system number received in
      the BGP OPEN message.";
    reference
```



```
    "RFC 4271, Section 4.2.";
  }
  leaf peer-as {
    type inet:as-number;
    description
      "AS number of the peer.";
  }
  leaf local-as {
    type inet:as-number;
    description
      "The local autonomous system number that is to be used when
      establishing sessions with the remote peer or peer group, if
      this differs from the global BGP router autonomous system
      number.";
  }

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

  container route-flap-damping {
    if-feature "bt:damping";
    leaf enable {
      type boolean;
      default "false";
      description
        "Enable route flap damping.";
    }
    leaf suppress-above {
      type decimal64 {
        fraction-digits 1;
      }
      default "3.0";
      description
        "This is the value of the instability metric at which
        route suppression takes place. A route is not installed
        in the forwarding information base (FIB), or announced
        even if it is reachable during the period that it is
        suppressed.";
    }
  }
  leaf reuse-above {
    type decimal64 {
```



```
        fraction-digits 1;
    }
    default "2.0";
    description
        "This is the value of the instability metric at which a
        suppressed route becomes unsuppressed if it is reachable
        but currently suppressed. The value assigned to
        reuse-below must be less than suppress-above.";
}
leaf max-flap {
    type decimal64 {
        fraction-digits 1;
    }
    default "16.0";
    description
        "This is the upper limit of the instability metric. This
        value must be greater than the larger of 1 and
        suppress-above.";
}
leaf reach-decay {
    type yang:gauge32;
    units "seconds";
    default "300";
    description
        "This value specifies the time desired for the instability
        metric value to reach one-half of its current value when
        the route is reachable. This half-life value determines
        the rate at which the metric value is decayed. A smaller
        half-life value makes a suppressed route reusable sooner
        than a larger value.";
}
leaf unreach-decay {
    type yang:gauge32;
    units "seconds";
    default "900";
    description
        "This value acts the same as reach-decay except that it
        specifies the rate at which the instability metric is
        decayed when a route is unreachable. It should have a
        value greater than or equal to reach-decay.";
}
leaf keep-history {
    type yang:gauge32;
    units "seconds";
    default "1800";
    description
        "This value specifies the period over which the route
        flapping history is to be maintained for a given route.
```



```
        The size of the configuration arrays described below is
        directly affected by this value.";
    }
    description
        "Routes learned via BGP are subject to weighted route
        dampening.";
}
leaf send-community {
    if-feature "bt:send-communities";
    type bt:community-type;
    description
        "When supported, this tells the router to propogate any
        prefixes that are attached to this community. The value
        of 0 implies 'none'.";
}
leaf description {
    type string;
    description
        "An optional textual description (intended primarily for use
        with a peer or group";
}
}

grouping neighbor-group-transport-config {
    description
        "Configuration parameters relating to the transport protocol
        used by the BGP session to the peer";
    leaf tcp-mss {
        type uint16;
        description
            "Sets the max segment size for BGP TCP sessions.";
    }
    leaf mtu-discovery {
        type boolean;
        default "false";
        description
            "Turns path mtu discovery for BGP TCP sessions on (true) or
            off (false)";
    }
    leaf passive-mode {
        type boolean;
        default "false";
        description
            "Wait for peers to issue requests to open a BGP session,
            rather than initiating sessions from the local router.";
    }
    leaf local-address {
        type union {
```



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

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

grouping graceful-restart-config {
  description
    "Configuration parameters relating to BGP graceful restart.";
  leaf enabled {
    type boolean;
    description
      "Enable or disable the graceful-restart capability.";
  }
  leaf restart-time {
    type uint16 {
      range "0..4096";
    }
    description
      "Estimated time (in seconds) for the local BGP speaker to
      restart a session. This value is advertise in the graceful
      restart BGP capability. This is a 12-bit value, referred to
      as Restart Time in RFC4724. Per RFC4724, the suggested
      default value is <= the hold-time value.";
    reference
      "RFC 4724: Graceful Restart Mechanism for BGP.";
  }
  leaf stale-routes-time {
    type uint32;
    description
      "An upper-bound on the time that stale routes will be
      retained by a router after a session is restarted. If an
      End-of-RIB (EOR) marker is received prior to this timer
      expiring stale-routes will be flushed upon its receipt - if
```



```
        no EOR is received, then when this timer expires stale paths
        will be purged. This timer is referred to as the
        Selection_Deferral_Timer in RFC4724";
    reference
        "RFC 4724: Graceful Restart Mechanism for BGP.";
}
leaf helper-only {
    type boolean;
    default "true";
    description
        "Enable graceful-restart in helper mode only. When this leaf
        is set, the local system does not retain forwarding its own
        state during a restart, but supports procedures for the
        receiving speaker, as defined in RFC4724.";
    reference
        "RFC 4724: Graceful Restart Mechanism for BGP.";
}
}

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



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

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



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

grouping state {
    description
        "Grouping containing common counters relating to prefixes and
        paths";
    leaf total-paths {
        type uint32;
        config false;
    }
}
```



```
        description
            "Total number of BGP paths 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@2020-06-28.yang"
submodule ietf-bgp-common-multiprotocol {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }

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

    // meta

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

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                Keyur Patel (keyur at arrcus.com),
                Susan Hares (shares at ndzh.com)";
    description
        "This sub-module contains groupings that are related to support
        for multiple protocols in BGP. The groupings are common across
        multiple contexts.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code. All rights reserved."
```


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

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

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

```
revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

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

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



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

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



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



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

// Common groupings across multiple AFI,SAFIs

grouping mp-all-afi-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 bt:percentage;
    description
      "Threshold on number of prefixes that can be received from
        a neighbour before generation of warning messages or log
        entries. Expressed as a percentage of max-prefixes";
  }
  leaf restart-timer {
    type uint32;
    units "seconds";
    description
      "Time interval in seconds after which the BGP session is
        re-established after being torn down due to exceeding the
        max-prefix limit.";
  }
}
}

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

grouping mp-l3vpn-ipv4-ipv6-unicast-common {
  description
    "Common configuration applied across L3VPN for IPv4
      and IPv6";
```



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

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

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

// Config groupings for common groups

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

```
<CODE BEGINS> file "ietf-bgp-common-structure@2020-06-28.yang"
submodule ietf-bgp-common-structure {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }

    import ietf-routing-policy {
        prefix rpol;
        reference
            "RFC ZZZZ, A YANG Data Model for Routing Policy Management";
    }
    import ietf-bgp-types {
        prefix bt;
```



```
reference
  "RFC XXXX, BGP YANG Model for Service Provider Network.";
}
include ietf-bgp-common-multiprotocol;
include ietf-bgp-common;

// meta

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

  Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
           Keyur Patel (keyur at arrcus.com),
           Susan Hares (shares at ndzh.com),
           Jeffrey Haas (jhaas at pfrc.org).";
description
  "This sub-module contains groupings that are common across
  multiple BGP contexts and provide structure around other
  primitive groupings.

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

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

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

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

revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
```



```
}

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

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

grouping structure-neighbor-group-ebgp-multihop {
  description
    "Structural grouping used to include eBGP multi-hop
    configuration and state for both BGP neighbors and peer
    groups";
  container ebgp-multihop {
    description
      "eBGP multi-hop parameters for the 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 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";
    reference
      "RFC 4456: BGP Route Reflection.";
    leaf route-reflector-cluster-id {
      when "../route-reflector-client = 'false'";
      type bt:rr-cluster-id-type;
      description
        "Route Reflector cluster id to use when local router is
        configured as a route reflector. Commonly set at the
        group level, but allows a different cluster id to be set
        for each neighbor.";
      reference
        "RFC 4456: BGP Route Reflection: An Alternative to
        Full Mesh.";
    }
    leaf no-client-reflect {
      type boolean;
      default "false";
      description
        "When set to 'true', this disables route redistribution
        by the Route Reflector. It is set 'true' when the client is
        fully meshed to prevent sending of redundant route
        advertisements.";
    }
    leaf route-reflector-client {
      type boolean;
      default "false";
      description
        "Configure the neighbor as a route reflector client.";
      reference
        "RFC 4456: BGP Route Reflection: An Alternative to
        Full Mesh.";
    }
  }
}

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



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

grouping structure-neighbor-group-add-paths {
  description
    "Structural grouping used to include ADD-PATHs configuration
    and state for both BGP neighbors and peer groups";
  container add-paths {
    if-feature "bt:add-paths";
    description
      "Parameters relating to the advertisement and receipt of
      multiple paths for a single NLRI (add-paths)";
    reference
      "RFC 7911: Advertisements of Multiple Paths in BGP.";
    leaf receive {
      type boolean;
      default "false";
      description
        "Enable ability to receive multiple path advertisements for
        an NLRI from the neighbor or group";
    }
    choice send {
      default "all";
      description
        "Choice of sending the max. number of paths or to send all.";
      case max {
        leaf max {
```



```
        type uint8;
        description
            "The maximum number of paths to advertise to neighbors
             for a single NLRI";
    }
}
case all {
    leaf all {
        type empty;
        description
            "Send all the path advertisements to neighbors for a
             single NLRI.";
    }
}
}
leaf eligible-prefix-policy {
    type leafref {
        path "/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@2020-06-28.yang"
submodule ietf-bgp-peer-group {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }

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

    // Include the common submodule
```



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

// meta

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

  Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
           Keyur Patel (keyur at arrcus.com),
           Susan Hares (shares at ndzh.com),
           Jeffrey Haas (jhaas at pfrc.org).";
description
  "This sub-module contains groupings that are specific to the
  peer-group context of the BGP module.

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

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

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

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

revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

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



```
description
  "List of address-families associated with the BGP peer-group";
list afi-safi {
  key "afi-safi-name";
  description
    "AFI, SAFI configuration available for the
     neighbour or group";
  uses mp-afi-safi-config;
  container graceful-restart {
    if-feature "bt:graceful-restart";
    description
      "Parameters relating to BGP graceful-restart";
    uses mp-afi-safi-graceful-restart-config;
  }
  uses route-selection-options;
  uses global-group-use-multiple-paths;
  uses mp-all-afi-safi-list-contents;
}
}

grouping bgp-peer-group-base {
  description
    "Parameters related to a BGP group.";
  leaf peer-group-name {
    type string;
    description
      "Name of the BGP peer-group";
  }
  uses neighbor-group-config;
  container timers {
    description
      "Timers related to a BGP peer-group.";
    uses neighbor-group-timers-config;
  }
  container transport {
    description
      "Transport session parameters for the BGP peer-group.";
    uses neighbor-group-transport-config;
  }
  container graceful-restart {
    if-feature "bt:graceful-restart";
    description
      "Parameters relating the graceful restart mechanism for BGP.";
    uses graceful-restart-config;
  }
  uses structure-neighbor-group-ebgp-multihop;
  uses structure-neighbor-group-route-reflector;
  uses structure-neighbor-group-as-path-options;
```



```
    uses structure-neighbor-group-add-paths;
    uses global-group-use-multiple-paths;
    uses rpol:apply-policy-group;
    container afi-safis {
        description
            "Per-address-family configuration parameters associated with
            the group.";
        uses bgp-peer-group-afi-safi-list;
    }
}

grouping bgp-peer-group-list {
    description
        "The list of BGP peer groups";
    list peer-group {
        key "peer-group-name";
        description
            "List of BGP peer-groups configured on the local system -
            uniquely identified by peer-group name";
        uses bgp-peer-group-base;
    }
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-neighbor@2020-06-28.yang"
submodule ietf-bgp-neighbor {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bgp;
    }

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

    // Include the common submodule

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

    // meta
```


organization

"IETF IDR Working Group";

contact

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

WG List: <idr@ietf.org>

Authors: Mahesh Jethanandani ([mjethanandani at gmail.com](mailto:mjethanandani@gmail.com)),
Keyur Patel ([keyur at arccus.com](mailto:keyur@arccus.com)),
Susan Hares ([shares at ndzh.com](mailto:shares@ndzh.com)),
Jeffrey Haas ([jhaas at pfrc.org](mailto:jhaas@pfrc.org)).";

description

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

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

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

Relating to IETF Documents

(<https://trustee.ietf.org/license-info>).

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

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

revision 2020-06-28 {

description

"Initial Version";

reference

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

}

grouping bgp-neighbor-use-multiple-paths {

description

"Multi-path configuration and state applicable to a BGP neighbor";

container use-multiple-paths {

description

"Parameters related to the use of multiple-paths for the same


```
        NLRI when they are received only from this neighbor";
    leaf enabled {
        type boolean;
        default "false";
        description
            "Whether the use of multiple paths for the same NLRI is
            enabled for the neighbor. This value is overridden by any
            more specific configuration value.";
    }
    container ebgp {
        description
            "Multi-path configuration for eBGP";
        leaf allow-multiple-as {
            type boolean;
            default "false";
            description
                "Allow multi-path to use paths from different neighboring
                ASes. The default is to only consider multiple paths
                from the same neighboring AS.";
        }
    }
}

grouping bgp-neighbor-counters-message-types-state {
    description
        "Grouping of BGP message types, included for re-use across
        counters";
    leaf updates-received {
        type uint64;
        description
            "Number of BGP UPDATE messages received from this neighbor.";
        reference
            "RFC 4273: bgpPeerInUpdates.";
    }
    leaf updates-sent {
        type uint64;
        description
            "Number of BGP UPDATE messages sent to this neighbor";
        reference
            "RFC 4273 - bgpPeerOutUpdates";
    }
    leaf messages-received {
        type uint64;
        description
            "Number of BGP messages received from thsi neighbor";
        reference
            "RFC 4273 - bgpPeerInTotalMessages";
    }
}
```



```
}
leaf messages-sent {
  type uint64;
  description
    "Number of BGP messages received from this neighbor";
  reference
    "RFC 4273 - bgpPeerOutTotalMessages";
}
leaf notification {
  type uint64;
  description
    "Number of BGP NOTIFICATION messages indicating an error
    condition has occurred exchanged.";
}
}

grouping bgp-neighbor-afi-safi-list {
  description
    "List of address-families associated with the BGP neighbor";
  list afi-safi {
    key "afi-safi-name";
    description
      "AFI, SAFI configuration available for the neighbor or
      group";
    uses mp-afi-safi-config;
    leaf active {
      type boolean;
      config false;
      description
        "This value indicates whether a particular AFI-SAFI has
        been successfully negotiated with the peer. An AFI-SAFI may
        be enabled in the current running configuration, but a
        session restart may be required in order to negotiate the
        new capability.";
    }
  }
  container prefixes {
    config false;
    description
      "Prefix counters for the BGP session";
    leaf received {
      type uint32;
      description
        "The number of prefixes received from the neighbor";
    }
    leaf sent {
      type uint32;
      description
        "The number of prefixes advertised to the neighbor";
    }
  }
}
```



```
    }
    leaf installed {
      type uint32;
      description
        "The number of advertised prefixes installed in the
        Loc-RIB";
    }
  }
  container graceful-restart {
    if-feature "bt:graceful-restart";
    description
      "Parameters relating to BGP graceful-restart";
    uses mp-afi-safi-graceful-restart-config;
    leaf received {
      type boolean;
      config false;
      description
        "This leaf indicates whether the neighbor advertised the
        ability to support graceful-restart for this AFI-SAFI";
    }
    leaf advertised {
      type boolean;
      config false;
      description
        "This leaf indicates whether the ability to support
        graceful-restart has been advertised to the peer";
    }
  }
  uses mp-all-afi-safi-list-contents;
  uses bgp-neighbor-use-multiple-paths;
}
}
}
<CODE ENDS>
```

[7.2.](#) BGP types

```
<CODE BEGINS> file "ietf-bgp-types@2020-06-28.yang"
module ietf-bgp-types {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-types";
  prefix bt;

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



```
// meta

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

  Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
           Keyur Patel (keyur at arrcus.com),
           Susan Hares (shares at ndzh.com),
           Jeffrey Haas (jhaas at pfrc.org).";
description
  "This module contains general data definitions for use in BGP
  policy. It can be imported by modules that make use of BGP
  attributes";

revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

/*
 * Features.
 */

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

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

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

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



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

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

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

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

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

/*
 * Identities.
 */

identity bgp-capability {
  description
    "Base identity for a BGP capability";
}

identity mp-bgp {
  base bgp-capability;
  description
```



```
    "Multi-protocol extensions to BGP";
  reference
    "RFC 4760: Multiprotocol Extensions for BGP-4.";
}

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

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

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

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

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



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

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

identity ipv4-labeled-unicast {
  base afi-safi-type;
  description
    "Labeled IPv4 unicast (AFI,SAFI = 1,4)";
  reference
    "RFC 3107: Carrying Label Information in BGP-4.";
}

identity ipv6-labeled-unicast {
  base afi-safi-type;
  description
    "Labeled IPv6 unicast (AFI,SAFI = 2,4)";
  reference
    "RFC 3107: Carrying Label Information in BGP-4.";
}

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

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



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

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

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

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

identity bgp-well-known-std-community {
  description
    "Base identity for reserved communities within the standard
    community space defined by RFC1997. These communities must
    fall within the range 0xFFFF0000 to 0xFFFFFFFF";
  reference
    "RFC 1997: BGP Communities Attribute.";
}

identity no-export {
  base bgp-well-known-std-community;
  description
    "Do not export NLRI received carrying this community outside
    the bounds of this autonomous system, or this confederation if
    the local autonomous system is a confederation member AS. This
```



```
        community has a value of 0xFFFFF01.";
    reference
        "RFC 1997: BGP Communities Attribute.";
}

identity no-advertise {
    base bgp-well-known-std-community;
    description
        "All NLRI received carrying this community must not be
        advertised to other BGP peers. This community has a value of
        0xFFFFF02.";
    reference
        "RFC 1997: BGP Communities Attribute.";
}

identity no-export-subconfed {
    base bgp-well-known-std-community;
    description
        "All NLRI received carrying this community must not be
        advertised to external BGP peers - including over confederation
        sub-AS boundaries. This community has a value of 0xFFFFF03.";
    reference
        "RFC 1997: BGP Communities Attribute.";
}

identity no-peer {
    base bgp-well-known-std-community;
    description
        "An autonomous system receiving NLRI tagged with this community
        is advised not to re-advertise the NLRI to external bi-lateral
        peer autonomous systems. An AS may also filter received NLRI
        from bilateral peer sessions when they are tagged with this
        community value";
    reference
        "RFC 3765: NOPEER Community for BGP.";
}

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

identity as-set {
    base as-path-segment-type;
    description
```



```
    "Unordered set of autonomous systems that a route in the UPDATE
      message has traversed.";
  reference
    "RFC 4271: A Border Gateway Protocol 4 (BGP-4), Section 4.3.";
}

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

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

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

/*
 * Typedefs.
 */

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



```
    description
      "Type to describe the direction of NLRI transmission";
  }

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

typedef bgp-std-community-type {
  type union {
    type uint32 {
      range "65536..4294901759";
    }
    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 {
  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 {
    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";
    }
    enum confederation {
      description
        "Confederation as peer";
    }
  }
  description
    "Labels a peer or peer group as explicitly internal,
    external or confederation.";
}
```

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

```
identity PRIVATE_AS_REMOVE_ALL {
```



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

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

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

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

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

typedef community-type {
    type bits {
```



```
    bit standard {
      position 0;
      description
        "Send only standard communities.";
      reference
        "RFC 1997: BGP Communities Attribute.";
    }
    bit extended {
      description
        "Send only extended communities.";
      reference
        "RFC 4360: BGP Extended Communities Attribute.";
    }
    bit large {
      description
        "Send only large communities.";
      reference
        "RFC 8092: BGP Large Communities Attribute.";
    }
  }
  description
    "Type describing variations of community attributes.
     The community types can be combined and a value of 0
     implies 'none'";
}
}
<CODE ENDS>
```

7.3. BGP policy data

```
<CODE BEGINS> file "ietf-bgp-policy@2020-06-28.yang"
module ietf-bgp-policy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-bgp-policy";
  prefix bp;

  // import some basic types

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



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

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

  Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
           Keyur Patel (keyur at arrcus.com),
           Susan Hares (shares at ndzh.com),
           Jeffrey Haas (jhaas at pfrc.org).";
description
  "This module contains data definitions for BGP routing policy.
  It augments the base routing-policy module with BGP-specific
  options for conditions and actions.";

revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXX, BGP Model for Service Provider Network.";
}

// typedef statements

typedef bgp-set-community-option-type {
  type enumeration {
    enum add {
      description
        "Add the specified communities to the existing
        community attribute";
    }
    enum remove {
      description
        "Remove the specified communities from the
        existing community attribute";
    }
    enum replace {
      description
        "Replace the existing community attribute with
        the specified communities. If an empty set is
        specified, this removes the community attribute
        from the route.";
    }
  }
}
```



```
    description
      "Type definition for options when setting the community
      attribute in a policy action";
  }

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

typedef bgp-set-med-type {
  type union {
    type uint32;
    type string {
      pattern '^([+-])([0-9]{1,8}|[0-3][0-9]{1,9}|4[0-1][0-9]{1,8}|'
        + '428[0-9]{1,7}|429[0-3][0-9]{1,6}|42948[0-9]{1,5}|'
        + '42949[0-5][0-9]{1,4}|429496[0-6][0-9]{1,3}|'
        + '4294971[0-9]{1,2}|42949728[0-9]|42949729[0-5])$';
    }
    type enumeration {
      enum igp {
        description
          "Set the MED value to the IGP cost toward the
          next hop for the route";
      }
      enum med-plus-igp {
        description
          "Before comparing MED values for path selection, adds to
          the MED the cost of the IGP route to the BGP next-hop
          destination.

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



```
        Useful when the downstream AS requires the complete
        cost of a certain route that is received across
        multiple ASs.";
    }
}
}
description
    "Type definition for specifying how the BGP MED can
    be set in BGP policy actions. The three choices are to set
    the MED directly, increment/decrement using +/- notation,
    and setting it to the IGP cost (predefined value).";
reference
    "None. WG needs to decide if this is going to be a standard.";
}

// augment statements

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



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

grouping set-community-action-common {
```



```
description
  "Common leaves for set-community and set-ext-community
  actions";
leaf method {
  type enumeration {
    enum inline {
      description
        "The extended communities are specified inline as a
        list";
    }
    enum reference {
      description
        "The extended communities are specified by referencing a
        defined ext-community set";
    }
  }
}
description
  "Indicates the method used to specify the extended
  communities for the set-ext-community action";
}
leaf options {
  type bgp-set-community-option-type;
  description
    "Options for modifying the community attribute with
    the specified values. These options apply to both
    methods of setting the community attribute.";
}
}

augment "/rpol:routing-policy/rpol:policy-definitions/" +
  "rpol:policy-definition/rpol:policy-statements/" +
  "rpol:statement/rpol:conditions" {
  description
    "BGP policy conditions added to routing policy module";
  container bgp-conditions {
    description
      "Top-level container for BGP specific policy conditions ";
    leaf med-eq {
      type uint32;
      description
        "Condition to check if the received MED value is equal to
        the specified value";
    }
    leaf origin-eq {
      type bt:bgp-origin-attr-type;
      description
        "Condition to check if the route origin is equal to the
        specified value";
    }
  }
}
```



```
}
leaf-list next-hop-in {
  type inet:ip-address-no-zone;
  description
    "List of next hop addresses to check for in the route
    update";
}
leaf-list afi-safi-in {
  type identityref {
    base bt:afi-safi-type;
  }
  description
    "List of address families which the NLRI may be within";
}
leaf local-pref-eq {
  type uint32;
  description
    "Condition to check if the local pref attribute is equal to
    the specified value";
}
leaf route-type {
  type enumeration {
    enum internal {
      description
        "route type is internal";
    }
    enum external {
      description
        "route type is external";
    }
  }
  description
    "Condition to check the route type in the route update";
}
container community-count {
  description
    "Value and comparison operations for conditions based on the
    number of communities in the route update";
}
container as-path-length {
  description
    "Value and comparison operations for conditions based on the
    length of the AS path in the route update.

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



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



```
}

augment "/rpol:routing-policy/rpol:policy-definitions/" +
    "rpol:policy-definition/rpol:policy-statements/" +
    "rpol:statement/rpol:actions" {
  description
    "BGP policy actions added to routing policy module.";
  container bgp-actions {
    description
      "Top-level container for BGP-specific actions";
    leaf set-route-origin {
      type bt:bgp-origin-attr-type;
      description
        "Set the origin attribute to the specified value";
    }
    leaf set-local-pref {
      type uint32;
      description
        "Set the local pref attribute on the route update";
    }
    leaf set-next-hop {
      type bgp-next-hop-type;
      description
        "Set the next-hop attribute in the route update";
    }
    leaf set-med {
      type bgp-set-med-type;
      description
        "Set the med metric attribute in the route update";
    }
    container set-as-path-prepend {
      description
        "Action to prepend local AS number to the AS-path a
        specified number of times";
      leaf repeat-n {
        type uint8 {
          range "1..max";
        }
        description
          "Number of times to prepend the local AS number to the AS
          path. The value should be between 1 and the maximum
          supported by the implementation.";
      }
    }
    container set-community {
      description
        "Action to set the community attributes of the route, along
        with options to modify how the community is modified.
```



```
    Communities may be set using an inline list OR
    reference to an existing defined set (not both).";
uses set-community-action-common;
container inline {
    when "../method = 'inline'" {
        description
            "Active only when the set-community method is inline";
    }
    description
        "Set the community values for the action inline with
        a list.";
    leaf-list communities {
        type union {
            type bt:bgp-std-community-type;
            type bt:bgp-well-known-community-type;
        }
        description
            "Set the community values for the update inline with a
            list.";
    }
}
container reference {
    when "../method = 'reference'" {
        description
            "Active only when the set-community method is reference";
    }
    description
        "Provide a reference to a defined community set for the
        set-community action";
    leaf community-set-ref {
        type leafref {
            path "/rpol:routing-policy/rpol:defined-sets/"
                + "bp:bgp-defined-sets/"
                + "bp:community-sets/bp:community-set/bp:name";
        }
        description
            "References a defined community set by name";
    }
}
}
container set-ext-community {
    description
        "Action to set the extended community attributes of the
        route, along with options to modify how the community is
        modified. Extended communities may be set using an inline
        list OR a reference to an existing defined set (but not
        both).";
    uses set-community-action-common;
```



```

container inline {
    when "../method = 'inline'" {
        description
            "Active only when the set-community method is inline";
    }
    description
        "Set the extended community values for the action inline
        with a list.";
    leaf-list communities {
        type union {
            type rt-types:route-target;
            type bt:bgp-well-known-community-type;
        }
        description
            "Set the extended community values for the update inline
            with a list.";
    }
}
container reference {
    when "../method = 'reference'" {
        description
            "Active only when the set-community method is reference";
    }
    description
        "Provide a reference to an extended community set for the
        set-ext-community action";
    leaf ext-community-set-ref {
        type leafref {
            path "/rpol:routing-policy/rpol:defined-sets/"
                + "bp:bgp-defined-sets/bp:ext-community-sets/"
                + "bp:ext-community-set/bp:name";
        }
        description
            "References a defined extended community set by name";
    }
}
}
}
}
}
}
<CODE ENDS>
```

7.4. RIB modules

```
<CODE BEGINS> file "ietf-bgp-rib@2020-06-28.yang"
submodule ietf-bgp-rib {
  yang-version 1.1;
  belongs-to ietf-bgp {
```



```
    prefix br;
  }

  /*
   * Import and Include
   */

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

  // groupings of attributes in three categories:
  // - shared across multiple routes
  // - common to LOC-RIB and Adj-RIB, but not shared across routes
  // - specific to LOC-RIB or Adj-RIB

  include ietf-bgp-rib-attributes;

  // groupings of annotations for each route or table

  include ietf-bgp-rib-table-attributes;

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

    Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
```



```
        Keyur Patel (keyur at arrcus.com),
        Susan Hares (shares at ndzh.com)";
description
  "Defines a submodule for representing BGP routing table (RIB)
  contents.  The submodule supports 5 logical RIBs per address
  family:

  loc-rib: This is the main BGP routing table for the local routing
  instance, containing best-path selections for each prefix. The
  loc-rib table may contain multiple routes for a given prefix,
  with an attribute to indicate which was selected as the best
  path. Note that multiple paths may be used or advertised even if
  only one path is marked as best, e.g., when using BGP
  add-paths. An implementation may choose to mark multiple
  paths in the RIB as best path by setting the flag to true for
  multiple entries.

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

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

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

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

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

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

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


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

```
revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network.";
}

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

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

    leaf origin {
      type bt:bgp-origin-attr-type;
      description
        "BGP attribute defining the origin of the path
        information.";
    }
    leaf atomic-aggregate {
      type boolean;
      description
        "BGP attribute indicating that the prefix is an atomic
        aggregate; i.e., the peer selected a less specific
        route without selecting a more specific route that is
        included in it.";
      reference
        "RFC 4271: Section 5.1.6.";
    }
    leaf next-hop {
      type inet:ip-address;
      description
        "BGP next hop attribute defining the IP address of the
        router that should be used as the next hop to the
        destination";
      reference
        "RFC 4271: Section 5.1.3.";
    }
    leaf med {
      type uint32;
```



```
    description
      "BGP multi-exit discriminator attribute used in BGP route
       selection process";
    reference
      "RFC 4271: Section 5.1.4.";
  }
  leaf local-pref {
    type uint32;
    description
      "BGP local preference attribute sent to internal peers to
       indicate the degree of preference for externally learned
       routes. The route with the highest local preference
       value is preferred.";
    reference
      "RFC 4271: Section 5.1.5.";
  }
  leaf originator-id {
    type yang:dotted-quad;
    description
      "BGP attribute that provides the id as an IPv4 address
       of the originator of the announcement.";
    reference
      "RFC 4456 - BGP Route Reflection: An Alternative to Full
       Mesh Internal BGP (IBGP)";
  }
  leaf-list cluster-list {
    type yang:dotted-quad;
    description
      "Represents the reflection path that the route has
       passed.";
    reference
      "RFC 4456 - BGP Route Reflection: An Alternative to Full
       Mesh Internal BGP (IBGP)";
  }
  leaf aigp-metric {
    type uint64;
    description
      "BGP path attribute representing the accumulated IGP
       metric for the path";
    reference
      "RFC 7311 - The Accumulated IGP Metric Attribute for BGP";
  }
  container aggregator {
    config false;
    description
      "BGP attribute indicating the prefix has been
       aggregated by the specified AS and router.";
    reference
```



```
    "RFC 4271: Section 5.1.7.";
  leaf as {
    type inet:as-number;
    description
      "AS number of the autonomous system that performed the
      aggregation.";
  }
  leaf as4 {
    type inet:as-number;
    description
      "AS number of the autonomous system that performed the
      aggregation (4-octet representation). This value is
      populated if an upstream router is not 4-octet capable.
      Its semantics are similar to the AS4_PATH optional
      transitive attribute";
    reference
      "RFC 6793 - BGP Support for Four-octet AS Number Space";
  }
  leaf address {
    type inet:ipv4-address;
    description
      "IP address of the router that performed the
      aggregation.";
  }
}
container as-path {
  description
    "Enclosing container for the list of AS path segments.

    In the Adj-RIB-In or Adj-RIB-Out, this list should show
    the received or sent AS_PATH, respectively. For
    example, if the local router is not 4-byte capable, this
    value should consist of 2-octet ASNs or the AS_TRANS
    (AS 23456) values received or sent in route updates.

    In the Loc-RIB, this list should reflect the effective
    AS path for the route, e.g., a 4-octet value if the
    local router is 4-octet capable.";
  reference
    "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)
    RFC 6793 - BGP Support for Four-octet AS Number Space
    RFC 5065 - Autonomous System Confederations for BGP";
  list segment {
    key "index";
    config false;
    uses bgp-as-path-attr;
    description
      "List of AS PATH segments";
```



```
    }
  }
  container as4-path {
    description
      "This is the path encoded with 4-octet
      AS numbers in the optional transitive AS4_PATH attribute.
      This value is populated with the received or sent
      attribute in Adj-RIB-In or Adj-RIB-Out, respectively.
      It should not be populated in Loc-RIB since the Loc-RIB
      is expected to store the effective AS-Path in the
      as-path leaf regardless of being 4-octet or 2-octet.";
    reference
      "RFC 6793 - BGP Support for Four-octet AS Number Space";
    list segment {
      key "index";
      config false;
      uses bgp-as-path-attr;
      description
        "List of AS PATH segments";
    }
  }
}

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

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

grouping attr-sets {
  description
    "A grouping for all sets of path attributes.";
```



```
    container attr-sets {
      description
        "Enclosing container for the list of path attribute sets";
      uses attr-set;
    }
  }

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

    leaf-list ext-community {
      type rt:route-target;
      description
        "List of BGP extended community attributes. The received
        extended community may be an explicitly modeled
        type or unknown, represented by an 8-octet value
        formatted according to RFC 4360.";
      reference
        "RFC 4360 - BGP Extended Communities Attribute";
    }
  }

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



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

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



```
        uses ipv4-adj-rib;
    }
    container ipv6-unicast {
        when "../afi-safi-name = 'bt:ipv6-unicast'" {
            description
                "Include this container for IPv6 unicast RIB";
        }
        description
            "Routing tables for IPv6 unicast -- active when the
             afi-safi name is ipv6-unicast";
        uses ipv6-loc-rib;
        uses ipv6-adj-rib;
    }
}
}
description
    "Top level container for BGP RIB";
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-rib-ext@2020-06-28.yang"
submodule ietf-bgp-rib-ext {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix bre;
    }

    include ietf-bgp-rib-types;

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

        Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
                 Keyur Patel (keyur at arrcus.com),
                 Susan Hares (shares at ndzh.com),
                 Jeffrey Haas (jhaas at pfrc.org).";
    description
        "Defines additional data nodes for the BGP RIB model.
         These items reflect extensions that are desirable features but
         are not currently supported in a majority of BGP
         implementations."
```


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

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

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

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

```
revision 2020-06-28 {
  description
    "Initial Revision.";
  reference
    "RFC XXXX: BGP YANG Model for Service Providers.";
}

grouping rib-ext-route-annotations {
  description
    "Extended annotations for routes in the routing tables";
  leaf reject-reason {
    type union {
      type identityref {
        base bgp-not-selected-bestpath;
      }
      type identityref {
        base bgp-not-selected-policy;
      }
    }
  }
  description
    "Indicates the reason the route is not used, either due to
    policy filtering or bestpath selection";
}
}
}
<CODE ENDS>
```



```
<CODE BEGINS> file "ietf-bgp-rib-types@2020-06-28.yang"
submodule ietf-bgp-rib-types {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix br;
  }

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

    Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
             Keyur Patel (keyur at arccus.com),
             Susan Hares (shares at ndzh.com),
             Jeffrey Haas (jhaas at pfrc.org).";
  description
    "Defines identity and type definitions associated with
    the BGP RIB modules.

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

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

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

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

  revision 2020-06-28 {
    description
      "Initial Version";
    reference
      "RFC XXXX, BGP Model for Service Provider Network.";
  }
}
```



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

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

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

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

identity invalid-confed {
  base invalid-route-reason;
  description
    "Route was invalid due to a loop in the AS_CONFED_SEQUENCE or
    AS_CONFED_SET attributes";
}

identity bgp-not-selected-bestpath {
  description
    "Base identity for indicating reason a route was was not
    selected by BGP route selection algorithm";
  reference
    "RFC 4271 - Section 9.1";
}

identity local-pref-lower {
  base bgp-not-selected-bestpath;
  description
    "Route has a lower localpref attribute than current best path";
  reference
    "RFC 4271 - Section 9.1.2";
}
```



```
}

identity as-path-longer {
  base bgp-not-selected-bestpath;
  description
    "Route has a longer AS path attribute than current best path";
  reference
    "RFC 4271 - Section 9.1.2.2 (a)";
}

identity origin-type-higher {
  base bgp-not-selected-bestpath;
  description
    "Route has a higher origin type, i.e., IGP origin is preferred
    over EGP or incomplete";
  reference
    "RFC 4271 - Section 9.1.2.2 (b)";
}

identity med-higher {
  base bgp-not-selected-bestpath;
  description
    "Route has a higher MED, or metric, attribute than the current
    best path";
  reference
    "RFC 4271 - Section 9.1.2.2 (c)";
}

identity prefer-external {
  base bgp-not-selected-bestpath;
  description
    "Route source is via IGP, rather than EGP.";
  reference
    "RFC 4271 - Section 9.1.2.2 (d)";
}

identity nexthop-cost-higher {
  base bgp-not-selected-bestpath;
  description
    "Route has a higher interior cost to the next hop.";
  reference
    "RFC 4271 - Section 9.1.2.2 (e)";
}

identity higher-router-id {
  base bgp-not-selected-bestpath;
  description
    "Route was sent by a peer with a higher BGP Identifier value,
```



```
        or router id";
    reference
        "RFC 4271 - Section 9.1.2.2 (f)";
}

identity higher-peer-address {
    base bgp-not-selected-bestpath;
    description
        "Route was sent by a peer with a higher IP address";
    reference
        "RFC 4271 - Section 9.1.2.2 (g)";
}

identity bgp-not-selected-policy {
    description
        "Base identity for reason code for routes that are rejected
        due to policy";
}

identity rejected-import-policy {
    base bgp-not-selected-policy;
    description
        "Route was rejected after apply import policies";
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-rib-attributes@2020-06-28.yang"
submodule ietf-bgp-rib-attributes {
    yang-version 1.1;
    belongs-to ietf-bgp {
        prefix br;
    }

    // import some basic types

    import ietf-bgp-types {
        prefix bgpt;
    }
    import ietf-inet-types {
        prefix inet;
    }
    include ietf-bgp-rib-types;

    // meta

    organization
```



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

  Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
           Keyur Patel (keyur at arrcus.com),
           Susan Hares (shares at ndzh.com)";
description
  "This submodule contains common data definitions for BGP
  attributes for use in BGP RIB tables.

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

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

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

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

revision 2020-06-28 {
  description
    "Initial version";
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Network";
}

grouping bgp-as-path-attr {
  description
    "Data for representing BGP AS-PATH attribute";
  leaf index {
    type uint16;
    description
      "An index into the AS segments.";
  }
  leaf type {
```



```
    type identityref {
      base bgpt:as-path-segment-type;
    }
    description
      "The type of AS-PATH segment";
  }
  leaf-list member {
    type inet:as-number;
    description
      "List of the AS numbers in the AS-PATH segment";
  }
}

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

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



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

grouping bgp-unknown-attr-state {
  description
    "Operational state data for path attributes not shared
     across route entries, common to LOC-RIB and Adj-RIB";
  leaf attr-type {
    type uint8;
    description
      "1-octet value encoding the attribute type code";
    reference
      "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
  }
  leaf attr-len {
    type uint16;
    description
      "One or two octet attribute length field indicating the
       length of the attribute data in octets. If the Extended
       Length attribute flag is set, the length field is 2 octets,
       otherwise it is 1 octet";
    reference
      "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)";
  }
}
```



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

grouping bgp-unknown-attr-top {
  description
    "Unknown path attributes that are not expected to be shared
    across route entries, common to LOC-RIB and Adj-RIB";
  container unknown-attributes {
    description
      "Unknown path attributes that were received in the UPDATE
      message which contained the prefix.";
    list unknown-attribute {
      key "attr-type";
      description
        "This list contains received attributes that are unrecognized
        or unsupported by the local router. The list may be empty.";
      uses bgp-unknown-attr-flags-state;
      uses bgp-unknown-attr-state;
    }
  }
}

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

grouping bgp-adj-rib-attr-state {
  description
    "Path attributes that are not expected to be shared across
    route entries, specific to Adj-RIB";
  leaf path-id {
    type uint32;
    description
      "When the BGP speaker supports advertisement of multiple
      paths for a prefix, the path identifier is used to
```


uniquely identify a route based on the combination of the prefix and path id. In the Adj-RIB-In, the path-id value is the value received in the update message. In the Loc-RIB, if used, it should represent a locally generated path-id value for the corresponding route. In Adj-RIB-Out, it should be the value sent to a neighbor when add-paths is used, i.e., the capability has been negotiated.";

reference

"[RFC 7911](#): Advertisement of Multiple Paths in BGP";

}

}

}

<CODE ENDS>

<CODE BEGINS> file "ietf-bgp-rib-table-attributes@2020-06-28.yang"

submodule ietf-bgp-rib-table-attributes {

yang-version 1.1;

belongs-to ietf-bgp {

prefix br;

}

// import some basic types

import ietf-yang-types {

prefix types;

reference

"[RFC 6991](#), Common YANG Data Types.";

}

include ietf-bgp-rib-types;

organization

"IETF IDR Working Group";

contact

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

WG List: <idr@ietf.org>

Authors: Mahesh Jethanandani (mjethanandani at gmail.com),

Keyur Patel (keyur at arccus.com),

Susan Hares (shares at ndzh.com);

description

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

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

Redistribution and use in source and binary forms, with or

without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in [Section 4.c](#) of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

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

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

```
revision 2020-06-28 {
  description
    "Initial version.";
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Network.";
}

grouping bgp-common-route-annotations-state {
  description
    "Data definitions for flags and other information attached
    to routes in both LOC-RIB and Adj-RIB";
  leaf last-modified {
    type types:timeticks;
    description
      "Timestamp when this path was last modified.

      The value is the timestamp in seconds relative to
      the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
  }
  leaf valid-route {
    type boolean;
    description
      "Indicates that the route is considered valid by the
      local router";
  }
  leaf invalid-reason {
    type identityref {
      base invalid-route-reason;
    }
    description
      "If the route is rejected as invalid, this indicates the
      reason.";
```



```
    }
  }

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

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

  grouping bgp-common-table-attrs-state {
    description
      "Common attributes attached to all routing tables";
    // placeholder for metadata associated with all tables
  }

  grouping bgp-common-table-attrs-top {
    // no enclosing container as this data will fit under an
    // existing LOC-RIB container
    uses bgp-common-table-attrs-state;
    description
      "Operational state data for data related to the entire
      LOC-RIB";
  }
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-rib-tables@2020-06-28.yang"
submodule ietf-bgp-rib-tables {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix br;
  }

  // import some basic types
```



```
import ietf-inet-types {
  prefix inet;
  reference
    "RFC 6991: Common YANG Data Types.";
}
import ietf-yang-types {
  prefix yang;
  reference
    "RFC 6991: Common YANG Data Types.";
}
import ietf-routing {
  prefix rt;
  reference
    "RFC 8022: A YANG Data Model for Routing Management";
}
import ietf-bgp-types {
  prefix bt;
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Network.";
}
include ietf-bgp-rib-ext;
include ietf-bgp-rib-attributes;
include ietf-bgp-rib-table-attributes;

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

  Editor: Mahesh Jethanandani (mjethanandani@gmail.com)
  Authors: Keyur Patel,
           Mahesh Jethanandani,
           Susan Hares";
description
  "This submodule contains structural data definitions for
   BGP routing tables.
```

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

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

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

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

```
revision 2020-06-28 {
  description
    "Initial Version";
  reference
    "RFC XXXX, BGP YANG Model for Service Provider Network.";
}

grouping bgp-adj-rib-common-attr-refs {
  description
    "Definitions of common references to attribute sets for
    multiple AFI-SAFIs for Adj-RIB tables";
  leaf attr-index {
    type leafref {
      path "../../../../../attr-sets/"
        + "attr-set/index";
    }
    description
      "Reference to the common attribute group for the
      route";
  }
  leaf community-index {
    type leafref {
      path "../../../../../communities/community/"
        + "index";
    }
    description
      "Reference to the community attribute for the route";
  }
  leaf ext-community-index {
    type leafref {
      path "../../../../../ext-communities/"
        + "ext-community/index";
    }
    description
      "Reference to the extended community attribute for the
      route";
  }
}
```



```
grouping bgp-loc-rib-common-attr-refs {
  description
    "Definitions of common references to attribute sets for
    multiple AFI-SAFIs for LOC-RIB tables";
  leaf attr-index {
    type leafref {
      path "../../../../../attr-sets/attr-set/"
        + "index";
    }
    description
      "Reference to the common attribute group for the
      route";
  }
  leaf community-index {
    type leafref {
      path "../../../../communities/community/"
        + "index";
    }
    description
      "Reference to the community attribute for the route";
  }
  leaf ext-community-index {
    type leafref {
      path "../../../../ext-communities/"
        + "ext-community/index";
    }
    description
      "Reference to the extended community attribute for the
      route";
  }
}

grouping bgp-loc-rib-common-keys {
  description
    "Common references used in keys for IPv4 and IPv6
    LOC-RIB entries";
  leaf origin {
    type union {
      type inet:ip-address;
      type identityref {
        base rt:routing-protocol;
      }
    }
    description
      "Indicates the origin of the route.  If the route is learned
      from a neighbor, this value is the neighbor address.  If
      the route was injected or redistributed from another
      protocol, the origin indicates the source protocol for the
```



```
        route.";
    }
    leaf path-id {
        type uint32;
        description
            "If the route is learned from a neighbor, the path-id
            corresponds to the path-id for the route in the
            corresponding adj-rib-in-post table. If the route is
            injected from another protocol, or the neighbor does not
            support BGP add-paths, the path-id should be set
            to zero, also the default value.

            However, YANG does not allow default values to be set
            for parameters that form the key, so a default value
            cannot be set here.";
    }
}

grouping clear-routes {
    description
        "Action to clear BGP routes.";
    container clear-routes {
        if-feature "bt:clear-routes";
        action clear {
            input {
                leaf clear-at {
                    type yang:date-and-time;
                    description
                        "The time, in the future when the clear operation will
                        be initiated.";
                }
            }
            output {
                leaf clear-finished-at {
                    type yang:date-and-time;
                    description
                        "The time when the clear operation finished.";
                }
            }
        }
        description
            "Action commands to clear routes governed by a if-feature.";
    }
}

grouping ipv4-loc-rib {
    description
        "Top-level grouping for IPv4 routing tables";
```



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

grouping ipv6-loc-rib {
  description
    "Top-level grouping for IPv6 routing tables";
  container loc-rib {
    config false;
    description
      "Container for the IPv6 BGP LOC-RIB data";
    uses bgp-common-table-attrs-top;
    container routes {
      description
        "Enclosing container for list of routes in the routing
```



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

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



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

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

grouping ipv4-adj-rib {
    description
        "Top-level grouping for Adj-RIB table";
    container neighbors {
```



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

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



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

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



```
        uses bgp-unknown-attr-top;
        uses rib-ext-route-annotations;
    }
}
}

grouping ipv6-adj-rib {
    description
        "Top-level grouping for Adj-RIB table";
    container neighbors {
        config false;
        description
            "Enclosing container for neighbor list";
        list neighbor {
            key "neighbor-address";
            description
                "List of neighbors (peers) of the local BGP speaker";
            leaf neighbor-address {
                type inet:ip-address;
                description
                    "IP address of the BGP neighbor or peer";
            }
        }
        container adj-rib-in-pre {
            description
                "Per-neighbor table containing the NLRI updates
                received from the neighbor before any local input
                policy rules or filters have been applied. This can
                be considered the 'raw' updates from the neighbor.";
            uses ipv6-adj-rib-common;
        }
        container adj-rib-in-post {
            description
                "Per-neighbor table containing the paths received from
                the neighbor that are eligible for best-path selection
                after local input policy rules have been applied.";
            uses ipv6-adj-rib-in-post;
        }
        container adj-rib-out-pre {
            description
                "Per-neighbor table containing paths eligible for
                sending (advertising) to the neighbor before output
                policy rules have been applied";
            uses ipv6-adj-rib-common;
        }
        container adj-rib-out-post {
            description
                "Per-neighbor table containing paths eligible for
                sending (advertising) to the neighbor after output
```



```
        policy rules have been applied";
    uses ipv6-adj-rib-common;
}
}
}
}
}
}
<CODE ENDS>
```

```
<CODE BEGINS> file "ietf-bgp-rib-table-attributes@2020-06-28.yang"
submodule ietf-bgp-rib-table-attributes {
  yang-version 1.1;
  belongs-to ietf-bgp {
    prefix br;
  }

  // import some basic types

  import ietf-yang-types {
    prefix types;
    reference
      "RFC 6991, Common YANG Data Types.";
  }
  include ietf-bgp-rib-types;

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

    Authors: Mahesh Jethanandani (mjethanandani at gmail.com),
             Keyur Patel (keyur at arccus.com),
             Susan Hares (shares at ndzh.com);

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

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

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


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

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

```
revision 2020-06-28 {
  description
    "Initial version.";
  reference
    "RFC XXXX: BGP YANG Model for Service Provider Network.";
}

grouping bgp-common-route-annotations-state {
  description
    "Data definitions for flags and other information attached
    to routes in both LOC-RIB and Adj-RIB";
  leaf last-modified {
    type types:timeticks;
    description
      "Timestamp when this path was last modified.

      The value is the timestamp in seconds relative to
      the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
  }
  leaf valid-route {
    type boolean;
    description
      "Indicates that the route is considered valid by the
      local router";
  }
  leaf invalid-reason {
    type identityref {
      base invalid-route-reason;
    }
    description
      "If the route is rejected as invalid, this indicates the
      reason.";
  }
}

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



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

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

    grouping bgp-common-table-attrs-state {
        description
            "Common attributes attached to all routing tables";
        // placeholder for metadata associated with all tables
    }

    grouping bgp-common-table-attrs-top {
        // no enclosing container as this data will fit under an
        // existing LOC-RIB container
        uses bgp-common-table-attrs-state;
        description
            "Operational state data for data related to the entire
            LOC-RIB";
    }
}
<CODE ENDS>
```

8. Contributors

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

9. Acknowledgements

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

Credit is also due to authors of the OpenConfig, whose model was relied upon to come up with this model.

Special thanks to Robert Wilton who helped convert the YANG models to a NMDA compatible model.

10. References

10.1. Normative references

- [RFC1997] Chandra, R., Traina, P., and T. Li, "BGP Communities Attribute", [RFC 1997](#), DOI 10.17487/RFC1997, August 1996, <<https://www.rfc-editor.org/info/rfc1997>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2439] Villamizar, C., Chandra, R., and R. Govindan, "BGP Route Flap Damping", [RFC 2439](#), DOI 10.17487/RFC2439, November 1998, <<https://www.rfc-editor.org/info/rfc2439>>.
- [RFC2918] Chen, E., "Route Refresh Capability for BGP-4", [RFC 2918](#), DOI 10.17487/RFC2918, September 2000, <<https://www.rfc-editor.org/info/rfc2918>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", [BCP 81](#), [RFC 3688](#), DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", [RFC 4271](#), DOI 10.17487/RFC4271, January 2006, <<https://www.rfc-editor.org/info/rfc4271>>.
- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.
- [RFC4456] Bates, T., Chen, E., and R. Chandra, "BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP)", [RFC 4456](#), DOI 10.17487/RFC4456, April 2006, <<https://www.rfc-editor.org/info/rfc4456>>.

- [RFC4659] De Clercq, J., Ooms, D., Carugi, M., and F. Le Faucheur, "BGP-MPLS IP Virtual Private Network (VPN) Extension for IPv6 VPN", [RFC 4659](#), DOI 10.17487/RFC4659, September 2006, <<https://www.rfc-editor.org/info/rfc4659>>.
- [RFC4724] Sangli, S., Chen, E., Fernando, R., Scudder, J., and Y. Rekhter, "Graceful Restart Mechanism for BGP", [RFC 4724](#), DOI 10.17487/RFC4724, January 2007, <<https://www.rfc-editor.org/info/rfc4724>>.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", [RFC 4760](#), DOI 10.17487/RFC4760, January 2007, <<https://www.rfc-editor.org/info/rfc4760>>.
- [RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", [RFC 4761](#), DOI 10.17487/RFC4761, January 2007, <<https://www.rfc-editor.org/info/rfc4761>>.
- [RFC5065] Traina, P., McPherson, D., and J. Scudder, "Autonomous System Confederations for BGP", [RFC 5065](#), DOI 10.17487/RFC5065, August 2007, <<https://www.rfc-editor.org/info/rfc5065>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", [RFC 5880](#), DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.
- [RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", [RFC 5881](#), DOI 10.17487/RFC5881, June 2010, <<https://www.rfc-editor.org/info/rfc5881>>.
- [RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", [RFC 5883](#), DOI 10.17487/RFC5883, June 2010, <<https://www.rfc-editor.org/info/rfc5883>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", [RFC 6020](#), DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](#), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", [RFC 6242](#), DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", [RFC 6514](#), DOI 10.17487/RFC6514, February 2012, <<https://www.rfc-editor.org/info/rfc6514>>.
- [RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", [RFC 6793](#), DOI 10.17487/RFC6793, December 2012, <<https://www.rfc-editor.org/info/rfc6793>>.
- [RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", [RFC 6811](#), DOI 10.17487/RFC6811, January 2013, <<https://www.rfc-editor.org/info/rfc6811>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", [RFC 6991](#), DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", [RFC 7911](#), DOI 10.17487/RFC7911, July 2016, <<https://www.rfc-editor.org/info/rfc7911>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", [RFC 7950](#), DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", [RFC 8177](#), DOI 10.17487/RFC8177, June 2017, <<https://www.rfc-editor.org/info/rfc8177>>.
- [RFC8277] Rosen, E., "Using BGP to Bind MPLS Labels to Address Prefixes", [RFC 8277](#), DOI 10.17487/RFC8277, October 2017, <<https://www.rfc-editor.org/info/rfc8277>>.

- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, [RFC 8341](#), DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", [RFC 8349](#), DOI 10.17487/RFC8349, March 2018, <<https://www.rfc-editor.org/info/rfc8349>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", [RFC 8446](#), DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8528] Bjorklund, M. and L. Lhotka, "YANG Schema Mount", [RFC 8528](#), DOI 10.17487/RFC8528, March 2019, <<https://www.rfc-editor.org/info/rfc8528>>.
- [RFC8529] Berger, L., Hopps, C., Lindem, A., Bogdanovic, D., and X. Liu, "YANG Data Model for Network Instances", [RFC 8529](#), DOI 10.17487/RFC8529, March 2019, <<https://www.rfc-editor.org/info/rfc8529>>.

10.2. Informative references

- [I-D.ietf-bfd-yang] Rahman, R., Zheng, L., Jethanandani, M., Pallagatti, S., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", [draft-ietf-bfd-yang-17](#) (work in progress), August 2018.
- [I-D.ietf-rtgwg-policy-model] Qu, Y., Tantsura, J., Lindem, A., and X. Liu, "A YANG Data Model for Routing Policy Management", [draft-ietf-rtgwg-policy-model-16](#) (work in progress), June 2020.
- [RFC3765] Huston, G., "NOPEER Community for Border Gateway Protocol (BGP) Route Scope Control", [RFC 3765](#), DOI 10.17487/RFC3765, April 2004, <<https://www.rfc-editor.org/info/rfc3765>>.
- [RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", [RFC 5082](#), DOI 10.17487/RFC5082, October 2007, <<https://www.rfc-editor.org/info/rfc5082>>.

- [RFC5925] Touch, J., Mankin, A., and R. Bonica, "The TCP Authentication Option", [RFC 5925](#), DOI 10.17487/RFC5925, June 2010, <<https://www.rfc-editor.org/info/rfc5925>>.
- [RFC7454] Durand, J., Pepelnjak, I., and G. Doering, "BGP Operations and Security", [BCP 194](#), [RFC 7454](#), DOI 10.17487/RFC7454, February 2015, <<https://www.rfc-editor.org/info/rfc7454>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", [RFC 8342](#), DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

[Appendix A](#). Examples

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

[A.1](#). Creating BGP Instance

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

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

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

[A.2.](#) Neighbor Address Family Configuration

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

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

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

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <routing
```



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



```
    </control-plane-protocols>
  </routing>
</config>
```

A.3. IPv6 Neighbor Configuration

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <key-chains
    xmlns="urn:ietf:params:xml:ns:yang:ietf-key-chain">
    <key-chain>
      <name>bgp-key-chain</name>
    </key-chain>
  </key-chains>
  <routing
    xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
    <control-plane-protocols>
      <control-plane-protocol>
        <type
          xmlns:bgp="urn:ietf:params:xml:ns:yang:ietf-bgp">bgp:bgp
        </type>
        <name>name:BGP</name>
        <bgp
          xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
          <global>
            <as>64496</as>
            <afi-safis>
              <afi-safi>
                <afi-safi-name
                  xmlns:bt=
                    "urn:ietf:params:xml:ns:yang:ietf-bgp-types">
                  bt:ipv6-unicast
                </afi-safi-name>
              </afi-safi>
            </afi-safis>
          </global>
          <neighbors>
            <neighbor>
              <remote-address>2001:db8::</remote-address>
              <enabled>true</enabled>
              <secure-session-enable>true</secure-session-enable>
```



```

    <secure-session>
      <ao-keychain>bgp-key-chain</ao-keychain>
    </secure-session>
    <peer-as>64497</peer-as>
    <description>"Peer Router B"</description>
    <timers>
      <hold-time>120</hold-time>
      <keepalive>70</keepalive>
    </timers>
    <afi-safis>
      <afi-safi>
        <afi-safi-name
          xmlns:bt=
            "urn:ietf:params:xml:ns:yang:ietf-bgp-types">
          bt:ipv6-unicast
        </afi-safi-name>
      </afi-safi>
    </afi-safis>
  </neighbor>
</neighbors>
</bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</config>

```

A.4. VRF Configuration

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

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

```

<?xml version="1.0" encoding="UTF-8"?>
<config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <network-instances
    xmlns="urn:ietf:params:xml:ns:yang:ietf-network-instance">
    <network-instance>
      <name>vrf-red</name>
      <vrf-root>
        <routing
          xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
          <router-id>192.0.2.1</router-id>
          <control-plane-protocols>
            <control-plane-protocol>
              <type

```



```

        xmlns:bgpm=
        "urn:ietf:params:xml:ns:yang:ietf-bgp">bgpm:bgp
    </type>
    <name>BGP</name>
    <bgp
        xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
        <global>
            <as>64496</as>
            <afi-safis>
                <afi-safi>
                    <afi-safi-name
                        xmlns:bt=
                            "urn:ietf:params:xml:ns:yang:ietf-bgp-types">
                            bt:ipv4-unicast
                    </afi-safi-name>
                </afi-safi>
            </afi-safis>
        </global>
    </bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</vrf-root>
</network-instance>
<network-instance>
    <name>vrf-blue</name>
    <vrf-root>
        <routing
            xmlns="urn:ietf:params:xml:ns:yang:ietf-routing">
            <router-id>192.0.2.2</router-id>
            <control-plane-protocols>
                <control-plane-protocol>
                    <type
                        xmlns:bgpm=
                            "urn:ietf:params:xml:ns:yang:ietf-bgp">bgpm:bgp
                    </type>
                    <name>BGP</name>
                    <bgp
                        xmlns="urn:ietf:params:xml:ns:yang:ietf-bgp">
                        <global>
                            <as>64496</as>
                            <afi-safis>
                                <afi-safi>
                                    <afi-safi-name
                                        xmlns:bt=
                                            "urn:ietf:params:xml:ns:yang:ietf-bgp-types">
                                            bt:ipv4-unicast
                                    </afi-safi-name>
                                </afi-safi>
                            </afi-safis>
                        </global>
                    </bgp>
                </control-plane-protocol>
            </control-plane-protocols>
        </routing>
    </vrf-root>
</network-instance>
</vrf-root>
</routing>

```



```
        </afi-safi>
      </afi-safis>
    </global>
  </bgp>
</control-plane-protocol>
</control-plane-protocols>
</routing>
</vrf-root>
</network-instance>
</network-instances>
</config>
```

Appendix B. How to add a new AFI and Augment a Module

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

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

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

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

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

  organization
    "Newafi model group.";
```



```
contact
  "abc@newafi.com";
description
  "This YANG module defines and uses new AFI.";

revision 2020-06-28 {
  description
    "Creating new AFI and using in this model";

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

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

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

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


[Appendix C](#). How to deviate a module

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

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

  import ietf-bgp {
    prefix "bgp";
  }

  organization
    "Newco model group.";

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

  revision 2020-06-28 {
    description
      "Creating NewCo deviations to ietf-bgp model";

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

  deviation "/bgp:bgp/bgp:global/bgp:graceful-restart/" +
    "bgp:restart-time" {
    deviate not-supported;
  }

  deviation "/bgp:bgp/bgp:global/bgp:graceful-restart/" +
    "bgp:stale-route-time" {
    deviate not-supported;
  }
}
```

Authors' Addresses

Mahesh Jethanandani
Kloud Services

Email: mjethanandani@gmail.com

Keyur Patel

Arcus

CA

USA

Email: keyur@arcus.com

Susan Hares

Huawei

7453 Hickory Hill

Saline, MI 48176

USA

Email: shares@ndzh.com

Jeffrey Haas

Juniper Networks

Email: jhaas@pfrc.org

