

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 30, 2015

A. Zhdankin
K. Patel
A. Clemm
Cisco
S. Hares
Huawei
M. Jethanandani
Ciena
X. Liu
Ericsson
January 26, 2015

Yang Data Model for BGP Protocol
draft-zhdankin-idr-bgp-cfg-00.txt

Abstract

This document defines a YANG data model that can be used to configure and manage BGP.

Status of This Memo

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

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

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

This Internet-Draft will expire on July 30, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
2. Definitions and Acronyms	4
3. The Design of the Core Routing Data Model	4
3.1. Overview	4
3.2. BGP Router Configuration	4
3.2.1. AF Configuration	5
3.2.1.1. AF Specific Protocol Configuration	7
3.2.1.2. BGP Bestpath Configuration	8
3.2.1.3. BGP Neighbor Configuration	8
3.2.1.4. BGP Dampening	8
3.2.1.5. BGP Route Aggregation	8
3.2.1.6. BGP Redistribution	8
3.2.2. BGP Neighbor Configuration	8
3.2.3. BGP RPKI	11
3.3. Prefix Lists	12
4. BGP Yang Module	12
5. IANA Considerations	42
6. Security Considerations	42
7. Acknowledgements	42
8. Contributors	42
9. References	42
9.1. Normative References	42
9.2. Informative References	43
Authors' Addresses	43

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. ReST) and encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage BGP. The data model is very comprehensive in scope, resulting in a very large module being defined. When contemplating whether it would be appropriate to introduce a data model of such a large scope, we decided that there would be value in particular because BGP defines such a rich set of features, which makes the problem arising from heterogeneity involved when managing these features quite pronounced. Also, there is very little information that is designated as "mandatory", leaving the decision which capabilities to actually support to product implementations.

There are several distinct parts of the data model. The first part, by far the largest, serves to configure and manage BGP itself. It defines a large set of control knobs for that purpose, as well as a few data nodes that can be used to monitor health and gather statistics. The second part, much smaller than the first, defines a data model for the configuration of AS-Path and prefix-based filter lists, in essence policies that define the exchange of BGP messages between BGP peers. Together they form a complete data model that serves as a framework for configuration and management of BGP protocol and its policies.

The YANG module defined in this document has all the common building blocks for BGP protocol namely: Neighbor List, Address Family specific Parameters, Protocol Bestpath specific Parameters, Prefix based Filter Lists, and AS-PATH based Filter Lists.

1.1. Requirements Language

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

2. Definitions and Acronyms

AF: Address Family

AS: Autonomous System

BGP: Border Gateway Protocol

HTTP: Hyper-Text Transfer Protocol

JSON: JavaScript Object Notation

L2VPN: Layer 2 VPN

NETCONF: Network Configuration Protocol

NSAP: Network Service Access Point

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

RPKI: Resource Public Key Infrastructure

RTFilter: Route Filter

VPN: Virtual Private Network

YANG: Data definition language for NETCONF

3. The Design of the Core Routing Data Model

3.1. Overview

The overall data model consists of two main components, each contained in its own separate container. Container "bgp-router" is used to configure and manage BGP itself. It is by far the largest part of the model. Container "prefix-lists" is used to configure BGP prefix lists. BGP prefix lists defines the rules and policies that helps BGP restrict information to share with which other nodes.

3.2. BGP Router Configuration

The overall structure of the "bgp-router" part of the model is depicted in the following diagram. Brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```

module: bgp
  +--rw bgp-router
  |   +--rw bgp-version?          string
  |   +--rw local-as-number?      uint32
  |   +--rw local-as-identifier?  inet:ip-address
  |   +--rw rpki-config
  |   |   .....
  |   +--rw router-id
  |   |   .....
  |   +--rw af-configuration
  |   |   .....
  +--rw bgp-neighbors
  |   .....

```

The key components of the "bgp-router" model concern the configuration of the BGP neighbors, of the Resource Public Key Infrastructure (RPKI), and of address families (AF). Each is defined in the following subsections.

3.2.1. AF Configuration

AF-configuration is used to configure and manage BGP configuration on an address family basis. BGP is designed to carry routing information for multiple different address families as specified in [RFC4760]. AF-Configuration is indexed by (router-AS, AFI, SAFI, VRFID) [RFC4760] and [RFC4364]. It contains any AF specific protocol configuration, BGP Bestpath configuration parameters, BGP neighbor configuration parameters, BGP dampening parameters, BGP route aggregation parameters, and any BGP policy configuration like redistribution.

The overall structure of the AF Configuration data model is depicted in the following diagram. As before, brackets enclose list keys, "rw" means configuration data, "?" designates optional nodes, parantheses indicate choices. The figure does not depict all definitions; it is intended to illustrate the overall model structure. Roughly speaking, address family configuration allows for separate configuration of IPv4, IPv6, L2VPN, NSAP, VPNv4 and VPNv6 address families, as well as route filters. Within each address family, you have additional substructure, for example, to distinguish between configuration of unicast and multicast.

```

module: bgp
  +--rw bgp-router
  |   .....
  |   +--rw af-configuration
  |   |   +--rw ipv4
  |   |   |   +--rw mdt

```

```

| .....
+--rw multicast
|   +--rw bgp
|   |   .....
|   +--rw auto-summary?          boolean
|   +--rw aggregate-address?     inet:ip-address
|   +--rw distance?              uint8
|   +--rw network?               inet:ip-address
|   +--rw (protocol)?
|   |   .....
|   +--rw default-metric?        uint32
+--rw unicast
|   +--rw bgp
|   |   .....
|   +--rw auto-summary?          boolean
|   +--rw aggregate-address?     inet:ip-address
|   +--rw distance?              uint8
|   +--rw network?               inet:ip-address
|   +--rw (protocol)?
|   |   .....
|   +--rw number-of-path?        uint8
|   +--rw ibgp-number-of-path?   uint8
|   +--rw synchronization?       boolean
+--rw mvpn
|   +--rw bgp
|   |   .....
|   +--rw auto-summary?          boolean
+--rw ipv6
|   +--rw multicast
|   |   +--rw bgp
|   |   |   .....
|   |   +--rw aggregate-address? inet:ip-address
|   |   +--rw distance?          uint8
|   |   +--rw network?           inet:ip-address
|   |   +--rw (protocol)?
|   |   |   .....
|   +--rw unicast
|   |   +--rw bgp
|   |   |   .....
|   |   +--rw aggregate-address? inet:ip-address
|   |   +--rw distance?          uint8
|   |   +--rw network?           inet:ip-address
|   |   +--rw (protocol)?
|   |   |   .....
|   |   +--rw default-metric?    uint32
|   |   +--rw number-of-path?    uint8
|   |   +--rw ibgp-number-of-path? uint8
|   |   +--rw synchronization?   boolean

```

```

|   |--rw mvpn
|   |   .....
|--rw l2vpn
|   |--rw vpls
|   |   .....
|--rw nsap
|   |--rw unicast
|   |   |--rw bgp
|   |   |   .....
|   |   |--rw default-metric?          uint32
|   |   |--rw number-of-path?          uint8
|   |   |--rw ibgp-number-of-path?     uint8
|   |   |--rw network?                  inet:ip-address
|   |   |--rw (protocol)?
|   |   |   .....
|   |   |--rw synchronization?         boolean
|--rw rtfilter
|   |--rw unicast
|   |   .....
|--rw vpv4
|   |--rw unicast
|   |   |--rw bgp
|   |   |   .....
|   |   |--rw number-of-path?          uint8
|   |   |--rw ibgp-number-of-path?     uint8
|--rw multicast
|   |--rw bgp
|   |   .....
|   |--rw number-of-path?          uint8
|   |--rw ibgp-number-of-path?     uint8
|--rw vpv6
|   |--rw unicast
|   |   |--rw bgp
|   |   |   .....

```

The key AF configuration components are described in the following subsections.

3.2.1.1. AF Specific Protocol Configuration

AF specific protocol configuration involves configuration of the parameters that are specific to a given AF. For instance, configuration parameters specific to the consistency checking between prefixes and labels are specific to address families that are enabled with Labels. Similarly redistribution of routes from other protocols is specific to Address Families that are supported in other protocols.

3.2.1.2. BGP Bestpath Configuration

BGP BestPath Configuration Parameters involves configuration of the parameters that influence the BGP Bestpath decision. For instance, the ignore-as-path command allows BGP process to ignore as-path length check. The ignore-routerid command allows BGP process to ignore routerid check. The ignore-igp-metric command allows BGP process to ignore igp metric check. The ignore-cost-community command allows BGP process to ignore cost communities. The MED related commands influence MED comparison in the BGP Bestpath decision.

3.2.1.3. BGP Neighbor Configuration

BGP Neighbor Configuration Parameters involves configuration of the parameters that are neighbor address family specific. These commands include neighbor capabilities, neighbor policies and any protocol related parameters that are specific to BGP neighbor.

3.2.1.4. BGP Dampening

BGP Dampening Parameters involves configuration of the parameters that influence BGP Route Dampening. These parameters allow enabling of Route Dampening on an address family level. The Dampening configuration also allows configuration of Dampening specific parameters like max suppress time, resuse threshold, half life, and the suppress threshold.

3.2.1.5. BGP Route Aggregation

BGP Route Aggregation Parameters involves configuration of the parameters that enables BGP Route Aggregation.

3.2.1.6. BGP Redistribution

BGP Route Redistribution Parameters involves configuration of the parameters that enables BGP Route Redistribution from and to the BGP protocol.

3.2.2. BGP Neighbor Configuration

Bgp-neighbor is used to configure and manage BGP neighbors. BGP neighbor configuration is indexed by af-configuration, neighbor address and neighbor-AS. It contains configuration for any policies that are configured for a neighbor on an inbound or an outbound, any transport related configuration parameters, any protocol related configuration parameters, and any protocol capabilities related configuration parameters.

BGP-neighbor-groups are used to configure and manage set of BGP neighbors with common configuration. BGP-neighbor-groups are indexed by af-configuration and group-name.

The following diagram depicts the overall structure of the BGP Neighbors subtree. Brackets enclose list keys, "rw" means configuration, "ro" operational state data, and "?" designates optional nodes. Parentheses enclose choice and case nodes. The figure does not depict all definitions; it is intended to illustrate the overall structure.

```

module: bgp
+ ....
+--rw bgp-neighbors
|   +--rw bgp-neighbor* [peer-address]
|   |   +--rw peer-address          inet:ip-address
|   |   +--rw remote-as             uint32
|   |   +--rw prefix-list?          prefix-list-ref
|   |   +--rw default-action?       actions-enum
|   |   +--rw af-specific-config
|   |   |   +--rw ipv4
|   |   |   |   +--rw mdt
|   |   |   |   |   .....
|   |   |   |   +--rw unicast
|   |   |   |   |   .....
|   |   |   |   +--rw multicast
|   |   |   |   |   .....
|   |   |   |   +--rw mvpn
|   |   |   |   |   .....
|   |   |   +--rw ipv6
|   |   |   |   +--rw unicast
|   |   |   |   |   .....
|   |   |   |   +--rw multicast
|   |   |   |   |   .....
|   |   |   |   +--rw mvpn
|   |   |   |   |   .....
|   |   |   +--rw l2vpn
|   |   |   |   +--rw evpn
|   |   |   |   |   .....
|   |   |   |   +--rw vpls
|   |   |   |   |   .....
|   |   |   +--rw nsap
|   |   |   |   +--rw unicast
|   |   |   |   |   .....
|   |   |   +--rw rtfilter
|   |   |   |   +--rw unicast
|   |   |   |   |   .....
|   |   |   +--rw vpnv4

```

```

| | | +--rw unicast
| | | | .....
| | | +--rw multicast
| | | | .....
| | +--rw vpnv6
| | | +--rw unicast
| | | | .....
| | | +--rw multicast
| | | | .....
+--rw session-open-mode?      enumeration
+--rw send-buffer-size?      uint32
+--rw receive-buffer-size?   uint32
+--rw precedence?            enumeration
+--rw tcp-mss?                uint16
+--rw ttl-security?          boolean
+--rw hold-time?              uint16
+--rw keepalive-time?        uint16
+--rw bgp-neighbor-state
| | .....
+--rw bgp-neighbor-statistics
| | .....
+--rw bgp-neighbor-groups
| +--rw bgp-neighbor-group* [nbr-grp-name]
| | +--rw nbr-grp-name          string
| | +--rw remote-as             uint32
| | +--rw prefix-list?         prefix-list-ref
| | +--rw default-action?      actions-enum
| | +--rw af-specific-config
| | | +--rw ipv4
| | | | +--rw mdt
| | | | | .....
| | | | +--rw unicast
| | | | | .....
| | | | +--rw multicast
| | | | | .....
| | | | +--rw mvpn
| | | | | .....
| | | +--rw ipv6
| | | | +--rw unicast
| | | | | .....
| | | | +--rw multicast
| | | | | .....
| | | | +--rw mvpn
| | | | | .....
| | | +--rw l2vpn
| | | | +--rw evpn
| | | | | .....
| | | +--rw vpls

```

```

| | | | | .....
| | | | | +--rw nsap
| | | | | | +--rw unicast
| | | | | | .....
| | | | | +--rw rtfilter
| | | | | | +--rw unicast
| | | | | | .....
| | | | | +--rw vpnv4
| | | | | | +--rw unicast
| | | | | | | .....
| | | | | | +--rw multicast
| | | | | | .....
| | | | | +--rw vpnv6
| | | | | | +--rw unicast
| | | | | | | .....
| | | | | | +--rw multicast
| | | | | | .....
| | | | | +--rw hold-time?                               uint16
| | | | | +--rw keepalive-time?                         uint16

```

3.2.3. BGP RPKI

rpki-config is used to configure and manage BGP Origin Validation. This feature is specific to IPv4 and IPv6 Address Families. It is indexed by af-configuration. It contains the configuration commands for the BGP RPKI Server, RPKI RTR Protocol and the BGP protocol. This includes configuration for the Server address, Server preference, RPKI RTR protocol specific parameters, choice of a transport for RPKI RTR Protocol, and BGP specific parameters including enabling and disabling of this feature for IBGP and EBGP routes.

The structure of the RPKI configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```

module: bgp
  +--rw bgp-router
  |
  | .....
  | +--rw rpki-config
  | |
  | | +--rw cache-server-config
  | | |
  | | | .....
  | | +--rw validation-config
  | | |
  | | | .....
  | | +--rw bestpath-computation
  | | |
  | | | .....

```

3.3. Prefix Lists

BGP Prefix Lists are used to manipulate Prefix information carried within a BGP. The prefix information carried within BGP is filtered or allowed using BGP Prefix Lists. BGP Prefix Lists consists of an ordered set of one or more rules that describe IPv4 or IPv6 prefixes range and an associated action rule that describes whether the matching prefixes should be dropped or permitted. The Prefix Lists are usually applied to a BGP neighbor as part of an inbound policy (applied to prefixes received by a neighbor) or an outbound policy (applied to prefixes sent by a neighbor).

The structure of the prefix list configuration data model is depicted below, per the same conventions used in the earlier diagrams.

```
module: bgp
.....
+--rw prefix-lists
  +--rw prefix-list [prefix-list-name]
    +--rw prefix-list-name    string
    +--rw prefixes
      +--rw prefix [seq-nr]
        +--rw seq-nr          uint16
        +--rw prefix-filter
          +--rw (ip-address-group)?
            | .....
            +--rw action          actions-enum
            +--rw statistics
              .....
              .....
```

Prefix lists are defined in a list in a designated container. Each prefix list in turn contains a list of prefixes, indexed by a sequency number. Each prefix is comprised of a prefix filter, used to match BGP packets, an action that is applied when a filter matches, and a set of statistics that indicate how often individual prefixes are applied.

4. BGP Yang Module

<CODE BEGINS> file "bgp@2013-07-15.yang"

```
module bgp {
  namespace "urn:cisco:params:xml:ns:yang:bgp";
  // replace with IANA namespace when assigned
  prefix bgp;

  import ietf-inet-types {
```

```
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }
  import ietf-routing {
    prefix routing;
    revision-date 2014-11-10;
  }

  organization
    "Cisco Systems
     170 West Tasman Drive
     San Jose, CA 95134-1706
     USA";
  contact
    "Alexander Clemm alex@cisco.com
     Keyur Patel keyupate@cisco.com
     Aleksandr Zhdankin azhdanki@cisco.com";
  description
    "This YANG module defines the generic configuration
     data for BGP, which is common across all of the vendor
     implementations of the protocol. It is intended that the module
     will be extended by vendors to define vendor-specific
     BGP configuration parameters and policies,
     for example route maps or route policies.

     Terms and Acronyms

     BGP (bgp): Border Gateway Protocol

     IP (ip): Internet Protocol

     IPv4 (ipv4): Internet Protocol Version 4

     IPv6 (ipv6): Internet Protocol Version 6

     MED(med): Multi Exit Discriminator

     IGP (igp): Interior Gateway Protocol

     MTU (mtu) Maximum Transmission Unit
     ";

  revision 2015-01-14 {
    description
      "Initial revision.";
```

```
    }

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

    typedef prefix-list-ref {
      description
        "A reference to the prefix list which a bgp-neighbor can use.";
      type leafref {
        path "/routing:routing/routing:routing-instance/routing:routing-protocols/
routing:routing-protocol/bgp:bgp-routing/bgp:prefix-lists/bgp:prefix-list/bgp:pr
efix-list-name";
      }
    }

    typedef bgp-peer-admin-status {
      description
        "Administrative status of a BGP peer.";
      type enumeration {
        enum "unknown";
        enum "up";
        enum "down";
      }
    }

    typedef actions-enum {
      description
        "Permit/deny action.";
      type enumeration {
        enum "permit";
        enum "deny";
      }
    }

    grouping ACTIONS {
      description
        "Permit/deny action.";
      leaf action {
        type actions-enum;
        mandatory true;
      }
    }

    grouping slow-peer-config {
      description
        "Configure a slow-peer.";
      container detection {
```

```
    leaf enable {
      type boolean;
      default "true";
    }
    leaf threshold {
      type uint16 {
        range "120..3600";
      }
    }
  }
  leaf split-update-group {
    type enumeration {
      enum "dynamic";
      enum "static";
    }
  }
}

grouping update-group-management {
  description
    "Manage peers in BGP update group.";
  leaf split-as-override {
    description
      "Keeps peers with as-override in different update groups.";
    type boolean;
  }
}

grouping neighbour-base-af-config {
  description
    "A set of configuration parameters that is applicable to all neighbour address families.";
  leaf active {
    description
      "Enable the address family for this neighbor.";
    type boolean;
    default "false";
  }
  leaf advertisement-interval {
    description
      "Minimum interval between sending BGP routing updates.";
    type uint32;
  }
  leaf allowas-in {
    description
      "Accept as-path with my AS present in it.";
    type boolean;
    default "false";
  }
}
```

```
leaf maximum-prefix {
  description
    "Maximum number of prefixes accepted from this peer.";
  type uint32;
}
leaf next-hop-self {
  description
    "Enable the next hop calculation for this neighbor.";
  type boolean;
  default "true";
}
leaf next-hop-unchanged {
  description
    "Propagate next hop unchanged for iBGP paths to this neighbour.";
  type boolean;
  default "true";
}

container remove-private-as {
  leaf remove-private-as-number {
    description
      "Remove private AS number from outbound updates.";
    type boolean;
  }
  leaf replace-with-local-as {
    description
      "Replace private AS number with local AS.";
    type boolean;
  }
}
leaf route-reflector-client {
  description
    "Configure a neighbor as Route Reflector client.";
  type boolean;
  default "false";
}
leaf send-community {
  description
    "Send Community attribute to this neighbor.";
  type enumeration {
    enum "both";
    enum "extended";
    enum "standard";
  }
  default "standard";
}
uses slow-peer-config;
leaf soo {
```



```
        description
          "Site-of-Origin extended community. Format is ASN:nn or IP-address:nn";
        type string;
      }
      leaf weight {
        description
          "Set default weight for routes from this neighbor.";
        type uint16;
      }
    }

    grouping neighbour-common-af-config {
      description
        "A set of configuration parameters that is applicable to all neighbour address families,
        except of nsap and rtfILTER.";
      uses neighbour-base-af-config;
      leaf prefix-list {
        description
          "Reference to the prefix list of this neighbour.";
        type prefix-list-ref;
      }
      leaf soft-reconfiguration {
        description
          "Allow inbound soft reconfiguration.";
        type boolean;
      }
    }

    grouping neighbour-cast-af-config {
      description
        "A set of configuration parameters that is applicable to both unicast and multicast sub-address families.";
      uses neighbour-common-af-config;
      leaf propagate-dmzlink-bw {
        description
          "Propagate the DMZ link bandwidth.";
        type boolean;
      }
      container default-originate {
        description
          "Originate default route to this neighbor.";
        leaf enable {
          type boolean;
          default "false";
        }
      }
    }

    grouping neighbour-ip-multicast-af-config {
```

```
    description
      "A set of configuration parameters that is applicable to ip multicast.";
    uses neighbour-cast-af-config;
    leaf route-server-client-context {
      description
        "Specifies Route Server client context name.";
      type string;
    }
  }

  grouping neighbour-ip-unicast-af-config {
    description
      "A set of configuration parameters that is applicable to ip unicast.
      This grouping is intended to be extended by vendors as necessary to describe the vendor-specific configuration parameters.";
    uses neighbour-ip-multicast-af-config;
  }

  grouping bgp-af-config {
    description
      "A set of configuration parameters that is applicable to all address families of the BGP router.";
    leaf additional-paths {
      description
        "Additional paths in the BGP table.";
      type enumeration {
        enum "all";
        enum "best-n";
        enum "group-best";
      }
    }
    leaf advertise-best-external {
      description
        "Advertise best external path to internal peers.";
      type boolean;
    }
    container aggregate-timer {
      description
        "Configure aggregation timer.";
      leaf enable {
        type boolean;
        default "true";
      }
      leaf threshold {
        type uint16 {
          range "6..60";
        }
      }
    }
  }
  container bestpath {
```

```

    description
      "Change the default bestpath selection.";
    choice bestpath-selection {
      case as-path {
        description
          "Configures a BGP router to not consider the autonomous system (AS)
path during best path route selection.";
        leaf ignore-as-path {
          type boolean;
          default "false";
        }
      }
      case compare-routerid {
        description
          "Configures a BGP router to compare identical routes received from
different external peers
          during the best path selection process and to select the route with
the lowest router ID as the best path.";
        leaf ignore-routerid {
          type boolean;
          default "false";
        }
      }
      case cost-community {
        description
          "Configures a BGP router to not evaluate the cost community attribut
e
          during the best path selection process.";
        leaf ignore-cost-community {
          type boolean;
          default "false";
        }
      }
      case igp-metric {
        description
          "Configures the system to ignore the IGP metric during BGP best path
selection.";
        leaf ignore-igp-metric {
          type boolean;
          default "false";
        }
      }
      case mad-confed {
        description
          "Configure a BGP routing process to compare the Multi Exit Discrimin
ator (MED)
          between paths learned from confederation peers.";
        leaf enable {
          type boolean;
          default "false";
        }
        leaf missing-as-worst {
          description
            "Assigns a value of infinity to routes that are missing

```

```

        the Multi Exit Discriminator (MED) attribute,
        making the path without a MED value the least desirable path";
        type boolean;
        default "false";
    }
}
}
}
uses bgp-dampening;
leaf propagate-dmzlink-bw {
    description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";
    type boolean;
}
leaf redistribute-internal {
    description
        "Allow redistribution of iBGP into IGPs (dangerous)";
    type boolean;
}
leaf scan-time {
    description
        "Configure background scanner interval in seconds.";
    type uint8 {
        range "5..60";
    }
}
uses slow-peer-config;
leaf soft-reconfig-backup {
    description
        "Use soft-reconfiguration inbound only when route-refresh is not negotia
ted.";
    type boolean;
}
}

grouping bgp-af-vpn-config {
    description
        "A set of configuration parameters that is applicable to vpn sub-address f
amily on the BGP router.";
    uses bgp-af-config;
    uses update-group-management;
}

grouping bgp-af-mvpn-config {
    description
        "A set of configuration parameters that is applicable to mvpn sub-address
family on the BGP router.";
    leaf scan-time {
        description
            "Configure background scanner interval in seconds.";
        type uint8 {

```

```
        range "5..60";
    }
}
uses slow-peer-config;
leaf soft-reconfig-backup {
    description
        "Use soft-reconfiguration inbound only when route-refresh is not negotia
ted.";
    type boolean;
}
leaf propagate-dmzlink-bw {
    description
        "Use DMZ Link Bandwidth as weight for BGP multipaths.";
    type boolean;
}
leaf rr-group {
    description
        "Extended community list name.";
    type string;
}
uses update-group-management;
}

grouping redistribute {
    description
        "Redistribute information from another routing protocol.
        This grouping is intended to be augmented by vendors to implement vendor-
specific protocol redistribution configuration options.";
    choice protocol {
        case bgp {
            leaf enable-bgp {
                type boolean;
            }
        }
        case ospf {
            leaf enable-ospf {
                type boolean;
            }
        }
        case isis {
            leaf enable-isis {
                type boolean;
            }
        }
        case connected {
            leaf enable-connected {
                type boolean;
            }
        }
        case eigrp {
```

```
        leaf enable-eigrp {
            type boolean;
        }
    }
    case mobile {
        leaf enable-mobile {
            type boolean;
        }
    }
    case static {
        leaf enable-static {
            type boolean;
        }
    }
    case rip {
        leaf enable-rip {
            type boolean;
        }
    }
}

grouping router-af-config {
    description
        "A set of configuration parameters that is applicable to all address families on the BGP router.";
    leaf aggregate-address {
        description
            "Configure BGP aggregate address.";
        type inet:ip-address;
    }
    leaf distance {
        description
            "Define an administrative distance.";
        type uint8 {
            range "1..255";
        }
    }
    leaf network {
        description
            "Specify a network to announce via BGP.";
        type inet:ip-address;
    }
    uses redistribute;
}

grouping maximum-paths {
    description
        "Configures packet forwarding over multiple paths.";
```

```
    leaf number-of-path {
      type uint8 {
        range "1..32";
      }
    }
    leaf ibgp-number-of-path {
      type uint8 {
        range "1..32";
      }
    }
  }
  grouping bgp-neighbor-config {
    leaf remote-as {
      type uint32;
      mandatory true;
    }
    leaf prefix-list {
      type prefix-list-ref;
    }
    leaf default-action {
      type actions-enum;
    }
  }

  leaf neighbor-group-name {
    description
      "Neighbor group name.";
    type string;
  }

  container af-specific-config {
    description
      "Address family specific configuration parameters for the neighbours.";
    container ipv4 {
      container mdt {
        uses neighbour-common-af-config;
      }
      container unicast {
        uses neighbour-ip-unicast-af-config;
      }
      container multicast {
        uses neighbour-ip-multicast-af-config;
      }
      container mvpn {
        uses neighbour-cast-af-config;
      }
    }
    container ipv6 {
      container unicast {
```

```
        uses neighbour-ip-unicast-af-config;
    }
    container multicast {
        uses neighbour-ip-multicast-af-config;
    }
    container mvpn {
        uses neighbour-common-af-config;
    }
}
container l2vpn {
    container evpn {
        uses neighbour-common-af-config;
    }
    container vpls {
        uses neighbour-common-af-config;
    }
}
container nsap {
    container unicast {
        uses neighbour-base-af-config;
        leaf prefix-list {
            type prefix-list-ref;
        }
    }
}
container rtfiler {
    container unicast {
        uses neighbour-base-af-config;
        leaf soft-reconfiguration {
            description
                "Allow inbound soft reconfiguration.";
            type boolean;
        }
    }
}
container vpnv4 {
    container unicast {
        uses neighbour-cast-af-config;
    }
    container multicast {
        uses neighbour-cast-af-config;
    }
}
container vpnv6 {
    container unicast {
        uses neighbour-cast-af-config;
    }
    container multicast {
```



```
        uses neighbour-cast-af-config;
    }
}
}

grouping bgp-neighbor-transport-config {
  leaf session-open-mode {
    description
      "Establish neighbor session using TCP Open mode.";
    type enumeration {
      enum "active";
      enum "passive";
    }
  }

  leaf send-buffer-size {
    description
      "Set socket BGP send buffer size.";
    type uint32;
  }

  leaf receive-buffer-size {
    description
      "Receive socket BGP send buffer size.";
    type uint32;
  }

  leaf precedence {
    description
      "Set Precedence.";
    type enumeration {
      enum "routine";
      enum "immediate";
      enum "flash";
      enum "flash-override";
      enum "critical";
      enum "internet";
      enum "network";
    }
  }

  leaf tcp-mss {
    description
      "TCP MSS.";
    type uint16;
  }
}
```

```
    leaf ttl-security {
      description
        "TTL Security.";
      type boolean;
    }
  }

  grouping bgp-neighbor-timers {
    leaf hold-time {
      description
        "BGP Hold Time interval.";
      default 180;
      type uint16;
    }

    leaf keepalive-time {
      description
        "BGP Keepalive Time interval.";
      default 60;
      type uint16;
    }
  }

  grouping bgp-dampening {
    container bgp-dampening-params {
      description
        "BGP Route Flap Dampening.";

      leaf half-time {
        description
          "Half Time for the penalty.";
        type uint8 {
          range "1..45";
        }
      }

      leaf Reuse-time {
        description
          "Reuse Time.";
        type uint16 {
          range "1..20000";
        }
      }

      leaf supresss-time {
        description
          "Supress Time.";
        type uint16 {
```

```
        range "1..20000";
      }
    }

    leaf max-supress-time {
      description
        "Max Supress Time";
      type uint8 {
        range "1..255";
      }
    }
  }
}

augment "/routing:routing/routing:routing-instance/routing:routing-protocols/routing:routing-protocol" {
  container bgp-routing {
    description
      "BGP routing configuration";
    must "/routing:routing/routing:routing-instance/routing:routing-protocols/routing:routing-protocol/routing:type = bgp-routing-protocol";
    container bgp-router {
      description
        "This is a top-level container for the BGP router.";
      leaf bgp-version {
        type string;
      }
      leaf local-as-number {
        type uint32;
      }
      leaf local-as-identifier {
        type inet:ip-address;
      }
    }
    container router-id {
      description
        "Configures a fixed router ID for the local BGP routing process.";
      leaf enable {
        type boolean;
      }
      choice config-type {
        case static {
          leaf ip-address {
            type boolean;
          }
        }
        case auto-config {
          leaf enable-auto-config {
            type boolean;
          }
        }
      }
    }
  }
}
```

```
    }

    container rpki-config {
      description
        "RPKI configuration parameters.";
      container cache-server-config {
        description
          "Configure the RPKI cache-server parameters in rpki-server configu
ration mode.";
        choice server {
          case ip-address {
            leaf ip-address {
              type inet:ip-address;
              mandatory true;
            }
          }
          case host-name {
            leaf ip-host-address {
              type inet:host;
              mandatory true;
            }
          }
        }
        choice transport {
          description
            "Specifies a transport method for the RPKI cache.";
          case tcp {
            leaf tcp-port {
              type uint32;
            }
          }
          case ssh {
            leaf ssh-port {
              type uint32;
            }
          }
        }
        leaf user-name {
          type string;
        }
        leaf password {
          type string;
        }
        leaf preference-value {
          description
            "Specifies a preference value for the RPKI cache.
            Setting a lower preference value is better.";
          type uint8 {
            range "1..10";
          }
        }
      }
    }
  }
}
```

```

    }
  }
  leaf purge-time {
    description
      "Configures the time BGP waits to keep routes from a cache after
the cache session drops. Set purge time in seconds.";
    type uint16 {
      range "30..360";
    }
  }
  choice refresh-time {
    description
      "Configures the time BGP waits in between sending periodic serial
queries to the cache. Set refresh-time in seconds.";
    case disable {
      leaf refresh-time-disable {
        type boolean;
      }
    }
    case set-time {
      leaf refresh-interval {
        type uint16 {
          range "15..3600";
        }
      }
    }
  }
}
choice response-time {
  description
    "Configures the time BGP waits for a response after sending a serial
or reset query. Set response-time in seconds.";
  case disable {
    leaf response-time-disable {
      type boolean;
    }
  }
  case set-time {
    leaf response-interval {
      type uint16 {
        range "15..3600";
      }
    }
  }
}
}
container validation-config {
  description
    "Controls the behavior of RPKI prefix validation processing.";
  leaf enable {
    description
      "Enables RPKI origin-AS validation.";
  }
}

```

```

        type boolean;
        default "true";
    }
    leaf enable-ibgp {
        description
            "Enables the iBGP signaling of validity state through an extende
d-community.";
        type boolean;
    }
    choice validation-time {
        description
            "Sets prefix validation time (in seconds) or to set off the auto
matic prefix validation after an RPKI update.";
        case validation-off {
            leaf disable {
                type boolean;
            }
        }
        case set-time {
            leaf prefix-validation-time {
                description
                    "Range in seconds.";
                type uint16 {
                    range "5..60";
                }
            }
        }
    }
}
container bestpath-computation {
    description
        "Configures RPKI bestpath computation options.";
    leaf enable {
        description
            "Enables the validity states of BGP paths to affect the path's p
reference in the BGP bestpath process.";
        type boolean;
    }
    leaf allow-invalid {
        description
            "Allows all 'invalid' paths to be considered for BGP bestpath co
mputation.";
        type boolean;
    }
}
uses bgp-neighbor-timers;
container af-configuration {
    description
        "Top level container for address families specific configuration of
the BGP router.";
    container ipv4 {
        container mdt {

```

```
        container bgp {
            description
                "BGP specific commands for ipv4-mdt address family/sub-address
family combination.";
            uses bgp-dampening;
            leaf scan-time {
                description
                    "Configure background scanner interval in seconds.";
                type uint8 {
                    range "5..60";
                }
            }
            uses slow-peer-config;
            leaf soft-reconfig-backup {
                description
                    "Use soft-reconfiguration inbound only when route-refresh is
not negotiated.";
                type boolean;
            }
            leaf propagate-dmzlink-bw {
                description
                    "Use DMZ Link Bandwidth as weight for BGP multipaths.";
                type boolean;
            }
        }
    }
    container multicast {
        container bgp {
            description
                "BGP specific commands for ipv4-multicast address family/sub-a
ddress family combination.";
            uses bgp-af-config;
        }
        leaf auto-summary {
            description
                "Enable automatic network number summarization";
            type boolean;
        }
        uses router-af-config;
        leaf default-metric {
            description
                "Set metric of redistributed routes.";
            type uint32;
        }
    }
    container unicast {
        container bgp {
            description
                "BGP specific commands for ipv4-unicast address family/sub-add
ess family combination.";
            uses bgp-af-config;
            leaf always-compare-med {
```

```
        description
            "Allow comparing MED from different neighbors.";
        type boolean;
        default "false";
    }
    leaf enforce-first-as {
        description
            "Enforce the first AS for EBGp routes(default).";
        type boolean;
        default "true";
    }
    leaf fast-external-fallover {
        description
            "Immediately reset session if a link to a directly connected
external peer goes down.";
        type boolean;
        default "true";
    }
    leaf suppress-inactive {
        description
            "Suppress routes that are not in the routing table.";
        type boolean;
    }
    leaf asnotation {
        description
            "Sets the default asplain notation.";
        type enumeration {
            enum "asplain";
            enum "dot";
        }
    }
    leaf enable-client-to-client-reflection {
        description
            "Manages client to client route reflection.";
        type boolean;
        default "true";
    }
    leaf cluster-id {
        description
            "Configure Route-Reflector Cluster-id.";
        type string;
    }
    container confederation {
        description
            "AS confederation parameters.";
        leaf identifier {
            description
                "Confederation identifier.";
            type string;
        }
    }
}
```



```

    }
    list peers {
      description
        "Confederation peers.";
      key "as-name";
      leaf as-name {
        type string;
      }
    }
  }
  container consistency-checker {
    description
      "Consistency-checker configuration.";
    leaf enable {
      type boolean;
    }
    leaf interval {
      description
        "Check interval in minutes.";
      type uint16 {
        range "5..1440";
      }
    }
    choice inconsistency-action {
      case error-message {
        description
          "Specifies that when an inconsistency is found, the syst
em will only generate a syslog message.";
        leaf generate-error-message-only {
          type boolean;
        }
      }
      case autorepair {
        description
          "Specifies that when an inconsistency is found,
the system will generate a syslog message and take acti
on
          based on the type of inconsistency found.";
        leaf perform-autorepair {
          type boolean;
        }
      }
    }
  }
  leaf deterministic-med {
    description
      "If enabled it enforce the deterministic comparison of the M
ED value between
      all paths received from within the same autonomous system."
    ;
    type boolean;
  }
}

```

```

        container graceful-restart {
            description
                "Controls the BGP graceful restart capability.";
            leaf enable {
                type boolean;
            }
            leaf restart-time {
                description
                    "Sets the maximum time period (in seconds) that the local
router will wait
                    for a graceful-restart-capable neighbor to return to norm
al operation after a restart event occurs.";
                type uint16 {
                    range "1..3600";
                }
                default "120";
            }
            leaf stalepath-time {
                description
                    "Sets the maximum time period that the local router will h
old stale paths for a restarting peer.";
                type uint16 {
                    range "5..3600";
                }
                default "360";
            }
        }
        container listener-config {
            description
                "Associates a subnet range with a BGP peer group and activat
e the BGP dynamic neighbors feature.";
            leaf enable {
                type boolean;
            }
            leaf limit {
                description
                    "Sets a maximum limit number of BGP dynamic subnet range n
eighbors.";
                type uint16 {
                    range "1..5000";
                }
                default "100";
            }
            leaf range {
                description
                    "Specifies a subnet range that is to be associated with a
specified peer group.";
                type uint16 {
                    range "0..32";
                }
            }
            leaf peer-group {
                description
                    "Specifies a BGP peer group that is to be associated with
the specified subnet range.";
            }
        }
    
```

```
        type string;
    }
}
leaf log-neighbor-changes {
    description
        "Log neighbor up/down and reset reason.";
    type boolean;
}
leaf max-as-limit {
    description
        "Configures BGP to discard routes that have a number of auto
nomous system numbers in AS-path that exceed the specified value.";
    type uint16 {
        range "1..254";
    }
}
container transport {
    description
        "Manages transport session parameters.";
    leaf enable-path-mtu-discovery {
        description
            "Enables transport path MTU discovery.";
        type boolean;
        default "true";
    }
}
}
leaf auto-summary {
    description
        "Enable automatic network number summarization";
    type boolean;
}
uses router-af-config;
uses maximum-paths;
leaf synchronization {
    description
        "Perform IGP synchronization.";
    type boolean;
}
}
container mvpn {
    container bgp {
        description
            "BGP specific commands for ipv4-mvpn address family/sub-address
family combination.";
        uses bgp-af-mvpn-config;
    }
    leaf auto-summary {
        description
            "Enable automatic network number summarization.";
    }
}
```

```
        type boolean;
      }
    }
  }
  container ipv6 {
    container multicast {
      container bgp {
        description
          "BGP specific commands for ipv6-multicast address family/sub-a
address family combination.";
        uses bgp-af-config;
      }
      uses router-af-config;
    }
    container unicast {
      container bgp {
        description
          "BGP specific commands for ipv6-unicast address family/sub-add
ess family combination.";
        uses bgp-af-config;
      }
      uses router-af-config;
      leaf default-metric {
        description
          "Set metric of redistributed routes.";
        type uint32;
      }
      uses maximum-paths;
      leaf synchronization {
        description
          "Perform IGP synchronization.";
        type boolean;
      }
    }
  }
  container mvpn {
    container bgp {
      description
        "BGP specific commands for ipv6-mvpn address family/sub-address
family combination.";
      uses bgp-af-mvpn-config;
    }
  }
}
container l2vpn {
  container vpls {
    container bgp {
      description
        "BGP specific commands for l2vpn-vpls address family/sub-addres
s family combination.";
      leaf scan-time {
        description
          "Configure background scanner interval in seconds.";
      }
    }
  }
}
```

```
        type uint8 {
            range "5..60";
        }
    }
    uses slow-peer-config;
}
}
container nsap {
    container unicast {
        container bgp {
            description
                "BGP specific commands for nsap-unicast address family/sub-add
ess family combination.";
            container aggregate-timer {
                description
                    "Configure Aggregation Timer.";
                leaf enable {
                    type boolean;
                    default "true";
                }
                leaf threshold {
                    type uint16 {
                        range "6..60";
                    }
                }
            }
        }
        uses bgp-dampening;
        leaf propagate-dmzlink-bw {
            description
                "Use DMZ Link Bandwidth as weight for BGP multipaths.";
            type boolean;
        }
        leaf redistribute-internal {
            description
                "Allow redistribution of iBGP into IGPs (dangerous)";
            type boolean;
        }
        leaf scan-time {
            description
                "Configure background scanner interval in seconds.";
            type uint8 {
                range "5..60";
            }
        }
        uses slow-peer-config;
        leaf soft-reconfig-backup {
            description
                "Use soft-reconfiguration inbound only when route-refresh is
not negotiated.";
```

```
        type boolean;
    }
}
leaf default-metric {
    description
        "Set metric of redistributed routes.";
    type uint32;
}
uses maximum-paths;
leaf network {
    description
        "Specify a network to announce via BGP.";
    type inet:ip-address;
}
uses redistribute;
leaf synchronization {
    description
        "Perform IGP synchronization.";
    type boolean;
}
}
}
container rtfiler {
    container unicast {
        container bgp {
            description
                "BGP specific commands for rtfiler-unicast address family/sub-
address family combination.";
            uses slow-peer-config;
        }
        uses maximum-paths;
    }
}
container vpnv4 {
    container unicast {
        container bgp {
            description
                "BGP specific commands for vpnv4-unicast address family/sub-ad-
dressed family combination.";
            uses bgp-af-vpn-config;
        }
        uses maximum-paths;
    }
    container multicast {
        container bgp {
            description
                "BGP specific commands for vpnv4-multicast address family/sub-
address family combination.";
            uses bgp-af-vpn-config;
        }
        uses maximum-paths;
    }
}
```

```

    }
  }
  container vpnv6 {
    container unicast {
      container bgp {
        description
          "BGP specific commands for vpnv6-unicast address family/sub-ad
address family combination.";
        uses bgp-af-vpn-config;
      }
    }
  }
}
container bgp-neighbors {
  description
    "The top level container for the list of neighbours of the BGP router.
";
  list bgp-neighbor {
    key "peer-address";
    leaf peer-address {
      type inet:ip-address;
      mandatory true;
    }
    uses bgp-neighbor-config;
    uses bgp-neighbor-transport-config;
    uses bgp-neighbor-timers;

    container bgp-neighbor-state {
      description
        "The operational parameters describing the neighbour state.
        It is intended that this container may be augmented by vendors to
reflect the vendor-specific operational state parameters.";
      leaf adminStatus {
        type bgp-peer-admin-status;
      }
      leaf in-lastupdatetime {
        type yang:timestamp;
      }
    }
    container bgp-neighbor-statistics {
      description
        "The operational parameters describing the neighbour statistics.
        It is intended that this container may be augmented by vendors to
reflect the vendor-specific staistical parameters.";
      leaf nr-in-updates {
        type uint32;
      }
      leaf nr-out-updates {
        type uint32;
      }
    }
  }
}

```

```
    }
  }
  container bgp-neighbor-groups {
    description
      "The top level container for the list of neighbour groups of the BGP r
outer.";
    list bgp-neighbor-group {
      key "nbr-grp-name";
      leaf nbr-grp-name {
        type string;
        mandatory true;
      }
      uses bgp-neighbor-config;
      uses bgp-neighbor-timers;
    }
  }
  container prefix-lists {
    description
      "Contains all prefix lists defined
on a router.";
    list prefix-list {
      key "prefix-list-name";
      description
        "A prefix list.";
      leaf prefix-list-name {
        type string;
      }
      container prefixes {
        list prefix {
          key "seq-nr";
          description
            "A prefix is a rule with a BGP filter.
The left hand side of the rule is the prefix filter.
It specifies a set of IP addresses.
If a BGP announcement contains an address that matches, the
rule is applied. The right hand side of the rule specifies
the action that is to be applied.";
          leaf seq-nr {
            type uint16;
            description
              "Sequence number of the rule.
The sequence number is included for compatibility purposes
with CLI; from a machine-to-machine interface perspective,
it would strictly speaking not be required as list elements
can be arranged in a particular order.";
          }
          container prefix-filter {
            choice ip-address-group {
              case ip-address {
```



```

        leaf ip-address {
            type inet:ip-address;
            mandatory true;
        }
    }
    case prefix {
        leaf prefix {
            type inet:ip-prefix;
            mandatory true;
        }
    }
    case host {
        leaf ip-host-address {
            type inet:host;
            mandatory true;
        }
    }
    case ip-range {
        leaf lower {
            type inet:ip-address;
        }
        leaf upper {
            type inet:ip-address;
        }
    }
}
leaf action {
    type actions-enum;
    mandatory true;
    description
        "permit/deny action";
}
container statistics {
    leaf prefix-hit-count {
        type uint32;
        config false;
    }
}
}
}
}
}
}
}
}
}
}
}
```

</CODE ENDS>

5. IANA Considerations

6. Security Considerations

The transport protocol used for sending the BGP data MUST support authentication and SHOULD support encryption. The data-model by itself does not create any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

7. Acknowledgements

The authors would like to thank Adam Simpson and Gunter Vandeveld for their detail reviews and comments.

8. Contributors

In addition to the authors listed on the front page, the following individuals have also helped to shape this document:

Dhanendra Jain

9. References

9.1. Normative References

- [I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Management",
draft-ietf-netmod-routing-cfg-15 (work in progress), May
2014.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629,
June 1999.
- [RFC2842] Chandra, R. and J. Scudder, "Capabilities Advertisement
with BGP-4", RFC 2842, May 2000.
- [RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC
Text on Security Considerations", BCP 72, RFC 3552, July
2003.
- [RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway
Protocol 4 (BGP-4)", RFC 4271, January 2006.

- [RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.
- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.
- [RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

9.2. Informative References

- [RFC5492] Scudder, J. and R. Chandra, "Capabilities Advertisement with BGP-4", RFC 5492, February 2009.

Authors' Addresses

Aleksandr Zhdankin
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: azhdanki@cisco.com

Keyur Patel
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Alexander Clemm
Cisco
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: alex@cisco.com

Susan Hares
Huawei
7453 Hickory Hill
Saline, MI 48176
USA

Email: shares@ndzh.com

Mahesh Jethanandani
Ciena
1741 Technology Drive
San Jose, CA 95110
USA

Email: mjethanandani@gmail.com

Xyfeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna, VA 22182
USA

Email: xufeng.liu@ericsson.com