Network Working Group
Internet-Draft
Intended status: Informational
Expires: June 30, 2015

G. Yan
S. Zhuang
Huawei Technologies

December 27, 2014

Yang Data Model for Routing Policy
draft-yan-rtgwg-routing-policy-yang-00

Abstract

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

Requirements Language

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

Status of This Memo

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

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

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

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

Copyright Notice

Table of Contents

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 intial confines, as bindings to other interfaces(e.g. ReST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and programatic APIs.

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

2. Definitions and Acronyms

ACL:Access Control List

AS: Autonomous System

BGP: Border Gateway Protocol

JSON: JavaScript Object Notation

NETCONF: Network Configuration Protocol

PBR:Policy-Based Routing

RD:Route Distinguisher

RPKI: Resource Public Key Infrastructure

VPN: Virtual Private Network

YANG: A data definition language for NETCONF

3. Introduction to Routing Policies

Routing policies are used to filter routes and set attributes for routes. Changing route attributes (including reachability) changes the path that network traffic passes through.

The difference between a routing policy and policy-based routing (PBR) is as follows:

o Routing policies apply to routes. Based on routing protocols, the result of route generation, advertisement, and selection is changed by following rules, changing parameters, or using control modes. That is, the contents in the routing table are changed.

o PBR applies to data packets. PBR provides a means to route or forward data packets flexibly based on predefined policies instead of following the routes in the existing routing table.

When advertising, receiving, and importing routes, the router implements certain policies based on actual networking requirements to filter routes and change the attributes of the routes. Routing policies serve the following purposes:

o Control route advertising: Only routes that match the rules specified in a policy are advertised.

o Control route receiving: Only the required and valid routes are received. This reduces the size of the routing table and improves network security.

o Filter and control imported routes: A routing protocol may import routes discovered by other routing protocols. Only routes that satisfy certain conditions are imported to meet the requirements of the protocol.

o Modify attributes of specified routes Attributes of the routes: that are filtered by a routing policy are modified to meet the requirements of the local device.

o Configure fast reroute (FRR): If a backup next hop and a backup outbound interface are configured for the routes that match a routing policy, IP FRR, VPN FRR, and IP+VPN FRR can be implemented.

Routing policies are implemented using the following procedures:

1) Define rules: Define features of routes to which routing policies are applied. Users define a set of matching rules based on different attributes of routes, such as the destination address and the address of the router that advertises the routes.

2) Implement the rules: Apply the matching rules to routing policies for advertising, receiving, and importing routes.

4. Design of Data Model

4.1. Overview

The routing policy Yang module is divided in following containers :

o asPathFilters : An AS_Path filter is used to filter BGP routes based on AS_Path attributes contained in the BGP routes..

o communityFilters : A community filter is used to filter BGP routes based on the community attributes contained in the BGP routes.

o extendCommunityFilters : An extcommunity filter is used to filter BGP routes based on extended community attributes.

o extendCommunitySooLists : A SoO extended community filter is used to filter BGP routes based on SoO extended community attributes.

o rdFilters : An RD filter is used to filter BGP routes based on RDs in VPN routes.

o prefixFilters : An IP prefix list contains a group of route filtering rules.

o ipv6PrefixFilters : An IPv6 prefix list contains a group of route filtering rules.

o routePolicys : A Route-Policy is a complex filter. It is used to match attributes of specified routes and change route attributes when

specific conditions are met. A Route-Policy can use the preceding filters to define its matching rules.

The figure below describe the overall structure of the routing policy Yang module :

```
module: routing-policy
   +--rw routing-policy
      +--rw asPathFilters
      ...
      +--rw communityFilters
      ...
      +--rw extendCommunityFilters
      ...
      +--rw extendCommunitySooLists
      ...
      +--rw rdFilters
      ...
      +--rw prefixFilters
      ...
      +--rw ipv6PrefixFilters
      ...
      +--rw routePolicys
         +--rw routePolicy* [name]
            +--rw name                string
            +--rw routePolicyNodes
               +--rw routePolicyNode* [nodeSequence]
                  +--rw nodeSequence      uint32
                  +--rw matchMode?        enumeration
                  +--rw description?      string
                  +--ro matchCount?       uint32
                  +--rw matchCondition
                  |  +--rw matchCostValue                 uint32
                  ...
                  |  +--rw matchExtCmntySooList?          string
                  +--rw applyAction
                     +--rw applyAsPaths
                     ...
                     +--rw applyPriorityValue           uint16
```

4.2. AS_Path Filter Configuration

An AS_Path filter uses the regular expression to define matching rules.

```
+--rw asPathFilters
|  +--rw asPathFilterKeyedByIndex* [index]
|  |  +--rw index                uint32
|  |  +--rw asPathFilterNodes
|  |     +--rw asPathFilterNode* [nodeSequence]
|  |        +--rw nodeSequence    uint32
|  |        +--rw matchMode?      enumeration
|  |        +--rw regular         string
|  +--rw asPathFilterKeyedByName* [name]
|     +--rw name                 string
|     +--rw asPathFilterNodes
|        +--rw asPathFilterNode* [nodeSequence]
|           +--rw nodeSequence    uint32
|           +--rw matchMode?      enumeration
|           +--rw regular         string
```

4.3. Community Filter Configuration

A community filter is used to filter BGP routes based on the community attributes contained in the BGP routes. The community attribute is a set of destination addresses with the same characteristics. Therefore, filtering rules defined based on community attributes can be used to filter BGP routes..

```
+--rw communityFilters
|  +--rw basicCommunityFilterKeyedByIndex* [index]
|  |  +--rw index                  uint32
|  |  +--rw basicCommunityNodes
|  |     +--rw basicCommunityNode* [nodeSequence]
...
|  +--rw basicCommunityFilterKeyedByName* [name]
|  |  +--rw name                   string
|  |  +--rw basicCommunityNodes
|  |     +--rw basicCommunityNode* [nodeSequence]
...
|  +--rw advancedCommunityFilterKeyedByIndex* [index]
...
|  +--rw advancedCommunityFilterKeyedByName* [name]
...
```

4.4. Extend Community Filter Configuration

An extendCommunity filter is used to filter BGP routes based on extended community attributes. An extcommunity filter is used to filter only BGP routes because the extended community attribute is a private attribute of BGP.

```
   +--rw extendCommunityFilters
   |  +--rw basicExtendCommunityFilterKeyedByIndex* [index]
   ..
   |  +--rw basicExtendCommunityFilterKeyedByName* [name]
   ...
   |  +--rw advancedExtendCommunityFilterKeyedByIndex* [index]
   ...
   |  +--rw advancedExtendCommunityFilterKeyedByName* [name]
   ...
```

4.5. Extend Community SoO Lists Configuration

An SoO extended community filter is used to filter BGP routes based on SoO extended community attributes.

```
   +--rw extendCommunitySooLists
   |  +--rw extendCommunitySooList* [name]
   |     +--rw filterType                              enumeration
   |     +--rw name                                    string
   |     +--rw basicExtendCommunitySooListNodes
   ...
   |     +--rw advanceExtendEommunitySooListNodes
   |        +--rw advanceExtendEommunitySooListNode* [nodeSequence matchMode]
   |           +--rw nodeSequence    uint32
   |           +--rw matchMode       enumeration
   |           +--rw regular         string
   ...
```

4.6. RD Filters Configuration

An RD filter is used to filter BGP routes based on RDs in VPN routes. RDs are used to distinguish IPv4 and IPv6 prefixes in the same address segment in VPN instances. RD filters specify matching rules regarding RD attributes.

```
   +--rw rdFilters
   |  +--rw rdFilter* [index]
   |     +--rw index            uint32
   |     +--rw rdFilterNodes
   |        +--rw rdFilterNode* [nodeSequence]
   |           +--rw nodeSequence    uint32
   |           +--rw matchMode?      enumeration
   |           +--rw rdStrings
   |              +--rw rdString* [rdStringValue]
   |                 +--rw rdStringValue    string
```

4.7. IPv4 Prefix Filters Configuration

An IP prefix list contains a group of route filtering rules. Users can specify the prefix and mask length range to match the destination network segment address or the next hop address of a route. An IP prefix list is used to filter routes that are advertised and received by various dynamic routing protocols..

```
+--rw prefixFilters
|  +--rw prefixFilter* [name]
|     +--rw name                  string
|     +--ro permitCnt?            uint32
|     +--ro denyCnt?              uint32
|     +--rw prefixFilterNodes
|        +--rw prefixFilterNode* [nodeSequence]
|           +--rw nodeSequence    uint32
|           +--rw matchMode?      enumeration
|           +--rw address         inet:ipv4-address
|           +--rw maskLength      uint8
|           +--rw matchNetwork?   boolean
|           +--rw greaterEqual?   uint8
|           +--rw lessEqual?      uint8
```

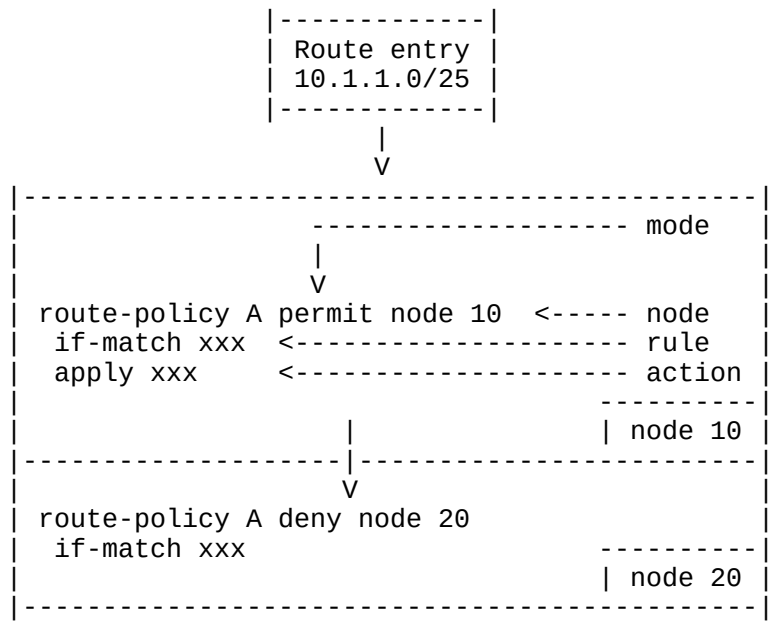
4.8. IPv6 Prefix Filters Configuration

An IPv6 prefix list contains a group of route filtering rules. Users can specify the prefix and mask length range to match the destination network segment address or the next hop address of a route. An IPv6 prefix list is used to filter routes that are advertised and received by various dynamic routing protocols..

```
+--rw ipv6PrefixFilters
|  +--rw ipv6PrefixFilter* [name]
|     +--rw name                      string
|     +--ro permitCnt?                uint32
|     +--ro denyCnt?                  uint32
|     +--rw ipv6PrefixFilterNodes
|        +--rw ipv6PrefixFilterNode* [nodeSequence]
|           +--rw nodeSequence    uint32
|           +--rw matchMode?      enumeration
|           +--rw address         inet:ipv6-address
|           +--rw maskLength      uint8
|           +--rw matchNetwork?   boolean
|           +--rw greaterEqual?   uint8
|           +--rw lessEqual?      uint8
```

4.9. Route-Policy Configuration

As shown in following figure, a Route-Policy consists of node IDs, matching mode, if-match clauses, and apply clauses.



Composition of a Route-Policy

o Node ID

A Route-Policy consists of one or more nodes. Node IDs are specified as indexes in the IP prefix list. In a Route-Policy, routes are filtered based on the following rules:

1) Sequential matching: The system checks entries based on node IDs in ascending order. Therefore, specifying the node IDs in the required sequence is recommended.

2) One-time matching: The relationship between the nodes of a Route-Policy is "OR". If a route matches one node, the route matches the Route-Policy and will not be matched against the next node.

o Matching mode

Either of the following matching modes can be used:

1) permit: specifies the permit mode of a node. If a route matches the if-match clauses of a node, all the actions defined by apply

clauses are performed, and the matching is complete. If a route does not match the if-match clauses of the node, the route continues to match the next node.

2) deny: specifies the deny mode of a node. In the deny mode, the apply clauses are not used. If a route matches all the if-match clauses of the node, the route is denied by the node and the next node is not matched. If the entry does not match all the if-match clauses, the next node is matched.

o if-match clause

The if-match clause defines the matching rules. Each node of a Route-Policy can comprise multiple if-match clauses or no if-match clause at all. If no if-match clause is configured for a node in the permit mode, all routes match the node.

o apply clause

The apply clauses specify actions. When a route matches a Route-Policy, the system sets some attributes for the route based on the apply clause. Each node of a Route-Policy can comprise multiple apply clauses or no apply clause at all. The apply clause is not used when routes need to be filtered but attributes of the routes do not need to be set.

Matching results of a Route-Policy

The matching results of a Route-Policy are obtained based on the following aspects:

1) Matching mode of the node, either permit or deny

2) Matching rules (either permit or deny) contained in the if-match clause (such as ACLs or IP prefix lists)

A Route-Policy is a complex filter. It is used to match attributes of specified routes and change route attributes when specific conditions are met. A Route-Policy can use the preceding filters to define its matching rules.

```
   +--rw routePolicys
      +--rw routePolicy* [name]
         +--rw name                string
         +--rw routePolicyNodes
            +--rw routePolicyNode* [nodeSequence]
               +--rw nodeSequence      uint32
               +--rw matchMode?        enumeration
               +--rw description?      string
               +--ro matchCount?       uint32
               +--rw matchCondition
               |  +--rw matchCostValue                 uint32
               ...
               |  +--rw matchExtCmntySooList?          string
               +--rw applyAction
                  +--rw applyAsPaths
                  ...
                  +--rw applyPriorityValue           uint16
```

5. Route Policy Yang Module

```
ROUTING POLICY YANG MODEL
<CODE BEGINS> file "routing-policy@2014-12-12.yang"
module routing-policy {
  namespace "urn:huawei:params:xml:ns:yang:routing-policy";
  // replace with IANA namespace when assigned
  prefix "routing-policy";

  import ietf-interfaces {
    prefix if;
    //rfc7223-YANG Interface Management
  }

  import ietf-inet-types {
    prefix inet;
    //rfc6991-Common YANG Data Types
  }

  description
    "This YANG module defines the RTP configuration data for
     routing policy service.

     Terms and Acronyms

     RTP: Routing-Policy
     Routing policies are used to filter routes and set attributes
     for routes. Changing route attributes (including reachability)
     changes the path that network traffic passes through.
```

```
    ";

  revision 2014-12-12 {
    description
      "Initial revision.";
  }

  grouping matchMode {
    leaf matchMode {
      config "true";
      type enumeration {
        enum "permit" {
          value "0";
          description "permit:
            Specifies the matching mode of the route-policy as permit.
            In permit mode, a route matches all the if-match clauses,
            the route matches the route-policy and the actions defined
            by the apply clause are performed on the route. Otherwise,
            the route continues to match the next entry.";
        }
        enum "deny" {
          value "1";
          description "deny:
            Specifies the matching mode of the route-policy as deny. In
            deny mode, if a route matches all the if-match clauses, the
            route fails to match the route-policy and cannot match the
            next node.";
        }
      }
    }
  }//End of grouping matchMode

  grouping asPathFilterNodes {
    container asPathFilterNodes {
      list asPathFilterNode {
        key "nodeSequence";
        min-elements "0";
        max-elements "unbounded";

        leaf nodeSequence {
          config "true";
          type uint32 {
            range "1..4294967295";
          }
        }
        uses matchMode;

        leaf regular {
```

```
          config "true";
          mandatory "true";
          type "string";
        }
      }
    }
  }//End of grouping asPathFilterNodes

  grouping basicCommunityNodes {
    container basicCommunityNodes {
      list basicCommunityNode {
        key "nodeSequence";
        min-elements "0";
        max-elements "unbounded";

        leaf nodeSequence {
          config "true";
          type uint32 {
            range "1..4294967295";
          }
        }
        uses matchMode;

        container communityNumbers {
          description
            "The ip community-filter basic comm-filter-name command or
             the ip community-filter basic-comm-filter-num command can
             be used to configure a basic community filter. basic
             comm-filter-name specifies the name of a basic community
             filter, and the name cannot be all digits. A maximum of 20
             community numbers can be configured using one command.
             basic-comm-filter-num specifies only the basic community
             filter with the number ranging from 1 to 99. A maximum of
             20 community numbers can be configured using one command.";
          list communityNumber {
            key "communityNumberValue";
            min-elements "0";
            max-elements "20";

            leaf communityNumberValue {
              config "true";
              type string;
            }
          }
        }

      }
    }
```

```
  }//End of grouping basicCommunityNodes

  grouping advancedCommunityNodes {
    container advancedCommunityNodes {

      list advancedCommunityNode {
        key "nodeSequence matchMode";
        min-elements "0";
        max-elements "unbounded";

        leaf nodeSequence {
          config "true";
          type uint32 {
            range "1..4294967295";
          }
        }
        uses matchMode;

        leaf regular {
          description
            "The ip community-filter advanced comm-filter-name command
             or the ip community-filter adv-comm-filter-num command can
             be used to configure an advanced community filter. advanced
             comm-filter-name specifies the name of an advanced
             community filter, and the name cannot be all digits.
             adv-comm-filter-num specifies only the advanced community
             filter with the number ranging from 100 to 199.";

          config "true";
          mandatory "true";
          type "string";
        }
      }

    }
  }//End of grouping advancedCommunityNodes

  grouping basicExtendCommunityFilterNodes {
    container extendCommunityFilterNodes {

      list extendCommunityFilterNode {
        key "nodeSequence";
        min-elements "0";
        max-elements "unbounded";

        leaf nodeSequence {
          config "true";
          type uint32 {
```

```
            range "1..4294967295";
          }
        }
        uses matchMode;
        container extendEommunityNumbers {
          list extendCommunityNumber {
            key "extendCommunityNumberValue";
            min-elements "0";
            max-elements "16";

            leaf extendCommunityNumberValue {
              config "true";
              type string;
            }
          }
        }
      }
    }
  }//End of grouping basicExtendCommunityFilterNodes

  grouping advancedExtendCommunityNodes {
    container advancedExtendCommunityNodes {

      list advancedExtendCommunityNode {

        key "nodeSequence matchMode";
        min-elements "0";
        max-elements "unbounded";

        leaf nodeSequence {
          config "true";
          type uint32 {
            range "1..4294967295";
          }
        }
        uses matchMode;

        leaf regular {
          config "true";
          mandatory "true";
          type "string";
        }
      }

    }
  }//End of grouping advancedExtendCommunityNodes

  container routing-policy {
```

```
    /*
      IP Routing Policy Configuration Commands.

      Routing policies are implemented using the following procedures:
      1)Define rules:
      Define features of routes to which routing policies are applied.
      Users define a set of matching rules based on different
      attributes of routes, such as the destination address and the
      address of the router that advertises the routes.
      2)Implement the rules:
      Apply the matching rules to routing policies for advertising,
      receiving, and importing routes.
      3)Filter
      A filter is the core of a routing policy and is defined using a
      set of matching rules.
     */
    container asPathFilters {
      description
        "An AS_Path filter uses the regular expression to define
         matching rules.";

      list asPathFilterKeyedByIndex {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "1..256";
          }
        }
        uses asPathFilterNodes;
      }

      list asPathFilterKeyedByName {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf name {
          config "true";
          type "string";
        }
        uses asPathFilterNodes;
      }
    } //End of container asPathFilters
```

```
    container communityFilters {
      description
        "The ip community-filter command configures a community filter
         or one entry in the community filter.

         The community attribute is a private attribute of BGP, and can
         be used only to filter BGP routes. The community attribute can
         be used together with the if-match community-filter command as
         a matching condition of a route-policy.";

      list basicCommunityFilterKeyedByIndex {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "1..99";
          }
        }
        uses basicCommunityNodes;
      }

      list basicCommunityFilterKeyedByName {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf name {
          config "true";
          type "string";
        }
        uses basicCommunityNodes;
      }

      list advancedCommunityFilterKeyedByIndex {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "100..199";
          }
        }
        uses advancedCommunityNodes;
```

```
      }

      list advancedCommunityFilterKeyedByName {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf name {
          config "true";
          type "string";
        }
        uses advancedCommunityNodes;
      }
    }//End of container communityFilters


    container extendCommunityFilters {
      description
        "The ip extcommunity-filter command adds an extended community
         filter.

         An extended community filter can be used as a matching
         condition of a route-policy by using a command such as
         if-match extcommunity-filter zz.";

      list basicExtendCommunityFilterKeyedByIndex {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "1..199";
          }
        }
        uses basicExtendCommunityFilterNodes;
      }
      list basicExtendCommunityFilterKeyedByName {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf name {
          config "true";
          type "string";
        }
        uses basicExtendCommunityFilterNodes;
```

```
      }
      list advancedExtendCommunityFilterKeyedByIndex {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "200..399";
          }
        }
        uses advancedExtendCommunityNodes;

      }
      list advancedExtendCommunityFilterKeyedByName {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf name {
          config "true";
          type "string";
        }
        uses advancedExtendCommunityNodes;
      }

    }//End of container extendCommunityFilters


    container extendCommunitySooLists {
      description
        "The ip extcommunity-list soo command configures a Source of
         Origin (SoO) extended community filter.

         SoO records the BGP route originator. To configure an SoO
         extended community filter so that BGP routes carrying SoO
         can be filtered, run the ip extcommunity-list soo command.";

      list extendCommunitySooList {
        key "name";
        min-elements "0";
        max-elements "unbounded";

        leaf filterType {
          config "true";
          mandatory "true";
          type enumeration {
```

```
            enum "basic" {
              value "0";
              description "basic:";
            }
            enum "advanced" {
              value "1";
              description "advanced:";
            }
          }
        }
        leaf name {
          config "true";
          type "string";
        }

        container basicExtendCommunitySooListNodes {
          list basicExtendCommunitySooListNode {
            key "nodeSequence";
            min-elements "0";
            max-elements "unbounded";

            leaf nodeSequence {
              config "true";
              type uint32 {
                range "1..4294967295";
              }
            }
            uses matchMode;

            container extendEommunityNumbers {
              list extendCommunityNumber {
                key "extendCommunityNumberValue";
                min-elements "1";
                max-elements "16";

                leaf extendCommunityNumberValue {
                  config "true";
                  type string;
                }
              }
            }
          }
        }

        container advanceExtendEommunitySooListNodes {
          list advanceExtendEommunitySooListNode {
            key "nodeSequence matchMode";
            min-elements "0";
```

```
            max-elements "unbounded";

            leaf nodeSequence {
              config "true";
              type uint32 {
                range "1..4294967295";
              }
            }

            uses matchMode;

            leaf regular {
              config "true";
              mandatory "true";
              type "string";
            }
          }

        }

      }
    }// End of container extendCommunitySooLists


    container rdFilters {
      description
        "The ip rd-filter command configures a route distinguisher (RD)
         filter.

         The RD attribute is carried in VPN routes. RDs are used to
         distinguish address spaces with the same IPv4 address prefix.
         An RD filter is used to filter VPN routes. The VPN target
         attribute of the VPN route controls the route exchange between
         VPN instances. The RD filter can filter a VPN route or multiple
         VPN routes from VPN instances.";

      list rdFilter {
        key "index";
        min-elements "0";
        max-elements "unbounded";

        leaf index {
          config "true";
          type uint32 {
            range "1..199";
          }
        }
```

```
        container rdFilterNodes {
          list rdFilterNode {
            key "nodeSequence";
            min-elements "0";
            max-elements "unbounded";

            leaf nodeSequence {
              config "true";
              type uint32 {
                range "1..4294967295";
              }
            }

            uses matchMode;

            container rdStrings {
              list rdString {
                key "rdStringValue";
                min-elements "0";
                max-elements "unbounded";

                leaf rdStringValue {
                  config "true";
                  type "string";
                }
              }

            }
          }
        }
      }//End of container rdFilters


      container prefixFilters {
        description
          "The ip ip-prefix command configures an new IP prefix list or
           one entry in an existing IP prefix list.

           An IP prefix list can be used as a filter or as matching
           conditions of a routing policy when it is used together with
           the if-match command.";

        list prefixFilter {
          key "name";
          min-elements "0";
          max-elements "unbounded";
```

```
        leaf name {
          config "true";
          type "string";
        }

        leaf permitCnt {
          config "false";
          default "0";
          type uint32 {
            range "0..65535";
          }
        }
        leaf denyCnt {
          config "false";
          default "0";
          type uint32 {
            range "0..65535";
          }
        }
        container prefixFilterNodes {
          list prefixFilterNode {
            key "nodeSequence";
            min-elements "0";
            max-elements "unbounded";

            leaf nodeSequence {
              config "true";
              type uint32 {
                range "1..4294967295";
              }
            }

            uses matchMode;

            leaf address {
              config "true";
              mandatory "true";
              type inet:ipv4-address;
            }
            leaf maskLength {
              config "true";
              mandatory "true";
              type uint8 {
                range "0..32";
              }
            }
            leaf matchNetwork {
              config "true";
```

```
            default "false";
            type "boolean";
          }
          leaf greaterEqual {
            config "true";
            type uint8 {
              range "0..32";
            }
          }
          leaf lessEqual {
            config "true";
            type uint8 {
              range "0..32";
            }
          }
        }
      }
    }
  }//End of container prefixFilters


  container ipv6PrefixFilters {
    description
      "The ip ipv6-prefix command configures an new IPv6 prefix list
       or one entry in an existing IPv6 prefix list.

       The IPv6 prefix list can be used by the protocols as a prefix
       filter, or used with the if-match ipv6 command as a matching
       condition of routing policies.";

    list ipv6PrefixFilter {

      key "name";
      min-elements "0";
      max-elements "unbounded";

      leaf name {
        config "true";
        type "string";
      }

      leaf permitCnt {
        config "false";
        default "0";
        type uint32 {
          range "0..65535";
        }
      }
```

```
        leaf denyCnt {
          config "false";
          default "0";
          type uint32 {
            range "0..65535";
          }
        }
        container ipv6PrefixFilterNodes {
          list ipv6PrefixFilterNode {
            key "nodeSequence";
            min-elements "0";
            max-elements "unbounded";

            leaf nodeSequence {
              config "true";
              type uint32 {
                range "1..4294967295";
              }
            }

            uses matchMode;

            leaf address {
              config "true";
              mandatory "true";
              type inet:ipv6-address;
            }
            leaf maskLength {
              config "true";
              mandatory "true";
              type uint8 {
                range "0..128";
              }
            }
            leaf matchNetwork {
              config "true";
              default "false";
              type "boolean";
            }
            leaf greaterEqual {
              config "true";
              type uint8 {
                range "0..128";
              }
            }
            leaf lessEqual {
              config "true";
              type uint8 {
```

```
              range "0..128";
            }
          }
        }

      }

    }
  }//End of container ipv6PrefixFilters

  /*
  Routing policies are used to filter routes and set attributes for
  routes. Changing route attributes (including reachability) changes
  the path that network traffic passes through.
  */
  container routePolicys {
    list routePolicy {
      key "name";
      min-elements "0";
      max-elements "unbounded";

      leaf name {
        config "true";
        type "string";
      }

      /*
      A route-policy is used to filter routes and set route attributes
      for the routes that match the route-policy. A route-policy
      consists of multiple nodes. One node can be configured with
      multiple if-match and apply clauses.
      */
      container routePolicyNodes {
        list routePolicyNode {
          key "nodeSequence";
          min-elements "0";
          max-elements "unbounded";

          leaf nodeSequence {
            config "true";
            type uint32 {
              range "0..65535";
            }
          }

          uses matchMode;

          leaf description {
```

```
            description
              "The description command configures the description of
               a route-policy.";
            config "true";
            mandatory "false";
            type "string";
          }

          leaf matchCount {
            config "false";
            mandatory "false";
            type "uint32";
          }

          /*if-match clauses*/
          container matchCondition {
            description
              "Define a set of matching rules and setting rules.
               The policy is applied to the routing information to
               meet the requirements of the matching rules.";

            leaf matchCostValue {
              description
                "The if-match cost command sets a matching rule that
                 is based on the route cost.

                 You can use the if-match cost command to configure
                 a node to filter routes based on the route costs.
                 After such a matching rule is configured, you can
                 apply the apply clauses to change the attributes of
                 the routes that match the matching rule.";

              config "true";
              mandatory "true";
              type uint32 {
                range "0..4294967295";
              }
            }//End of leaf matchCostValue

            container matchInterfaces {
              description
                "The if-match interface command sets a matching rule
                 that is based on the outbound interface.

                 The if-match interface command is used to filter
                 routes based on the outbound interfaces.
                 A maximum of 16 outbound interfaces can be
                 configured in this command. ";
```

```
                list matchInterface {
                  key "ifName";
                  min-elements "0";
                  max-elements "unbounded";

                  leaf ifName {
                    config "true";
                    type leafref {
                      path "/if:interfaces/if:interface/if:name";
                    }
                  }
                }
              }//End of container matchInterfaces

              container matchRouteTypes {
                description
                  "The if-match route-type command sets a filtering
                   rule that is based on the route type.

                   To filter OSPF or IS-IS routes based on the route
                   type, run the if-match route-type command.";

                list matchRouteType {
                  key "routeType";
                  min-elements "0";
                  max-elements "unbounded";

                  leaf routeType {
                    config "true";
                    type enumeration {
                      enum "external1" {
                        value "0";
                        description "external1:";
                      }
                      enum "external2" {
                        value "1";
                        description "external2:";
                      }
                      enum "internal" {
                        value "2";
                        description "internal:";
                      }
                      enum "isisLevel1" {
                        value "3";
                        description "isisLevel1:";
                      }
                      enum "isisLevel2" {
                        value "4";
```

```
                description "isisLevel2:";
              }
              enum "nssaExternal1" {
                value "5";
                description "nssaExternal1:";
              }
              enum "nssaExternal2" {
                value "6";
                description "nssaExternal2:";
              }
            }
          }
        }
      }//End of container matchRouteTypes

      leaf matchTagValue {
        description
          "The if-match tag command sets a matching rule
           that is based on the route tag.

           You can run the if-match tag command to filter
           OSPF or IS-IS routes based on the tags.";

        config "true";
        mandatory "true";
        type uint32 {
          range "0..4294967295";
        }
      }//End of leaf matchTagValue

      leaf matchMplsLabel {
        description
          "The if-match mpls-label command sets a matching
           rule that is based on MPLS labels. That is, a
           rule is set to match the routes with MPLS labels.

           In the scenario where inter-AS VPN Option C or
           CSC is deployed, you can use the if-match
           mpls-label command to match routes with MPLS
           labels.";

        config "true";
        mandatory "true";
        type "boolean";
      }

      leaf matchDestAcl {
        description
```

```
            "The if-match acl command sets a matching rule
             that is based on the Access Control List (ACL).";

          config "true";
          mandatory "true";
          type "string";
        }

        leaf matchDestPrefixFilter {
          description
            "The if-match ip-prefix command sets a matching
             rule that is based on the IP prefix list.";

          config "true";
          mandatory "true";
          type "string";
        }

        leaf matchDestPrefix6Filter {
          description
            "Specify the ipv6-prefix name to be matched";

          config "true";
          mandatory "true";
          type "string";
        }

        leaf matchDestAcl6 {
          description
            "Specify the ipv6 acl name or num.";

          config "true";
          mandatory "true";
          type "string";
        }

        container matchIPv4NextHop {
          description
            ".";

          choice matchIPv4NextHopMode {
            case matchNextHopPrefixFilter {
              leaf prefixName {
                config "true";
                mandatory "true";
                type "string";
              }
            }
```

```
                case matchNexthopAcl {
                  leaf aclNameOrNum {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
              }
            }

            container matchIPv6NextHop {
              choice matchIPv6NextHopMode {
                case matchNextHopPrefix6Filter {
                  leaf ipv6PrefixName {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
                case matchNextHopAcl6 {
                  leaf aclNameOrNum {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
              }
            }

            container matchIPv4RouteSource {
              choice matchIPv4RouteSourceMode {
                case matchRtSrcPrefixFilter {
                  leaf prefixName {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
                case matchRouteSourceAcl {
                  leaf aclNameOrNum {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
              }
            }
```

```
            container matchIPv6RouteSource {
              choice matchIPv6RouteSourceMode {
                case matchRtSrcPrefix6Filter {
                  leaf ipv6PrefixName {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
                case matchRtSrcAcl6 {
                  leaf aclNameOrNum {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
              }
            }

            container matchCommunityFilters {
              description
                "The if-match community-filter command sets a
                 matching rule that is based on the community
                 filter.";

              list matchCommunityFilter {

                key "cmntyNameOrNum";
                min-elements "0";
                max-elements "unbounded";

                leaf wholeMatch {
                  config "true";
                  default "false";
                  type "boolean";
                }
                leaf cmntyNameOrNum {
                  config "true";
                  type "string";
                }
              }
            }

            container matchExtcommunityFilters {
              description
                "The if-match extcommunity-filter command sets a
                 matching rule that is based on the extended
                 community filter.";
```

```
            list matchExtcommunityFilterIndex {
              key "extCmntyIndex";
              min-elements "0";
              max-elements "unbounded";

              leaf extCmntyIndex {
                config "true";
                type uint32 {
                  range "1..399";
                }
              }
            }
            list matchExtcommunityFilterName {

              key "extCmntyName";
              min-elements "0";
              max-elements "unbounded";

              leaf extCmntyName {
                config "true";
                type "string";
              }
            }
          }

          leaf matchRdFilter {
            description
              "The if-match rd-filter command sets a matching rule
               that is based on the Route Distinguisher (RD)
               filter.";

            config "true";
            mandatory "true";
            type uint32 {
              range "1..199";
            }
          }

          container matchAsPathFilters {
            description
              "The if-match as-path-filter command sets a matching
               rule that is based on the AS_Path filter.";

            list matchAsPathFilterIndex {
              key "asPathIndex";
              min-elements "0";
              max-elements "unbounded";
```

```
            leaf asPathIndex {
              config "true";
              type uint32 {
                range "1..256";
              }
            }
          }
          list matchAsPathFilterName {
            key "asPathFilterName";
            min-elements "0";
            max-elements "unbounded";

            leaf asPathFilterName {
              config "true";
              type "string";
            }
          }
        }

        leaf matchOriginAsValidateResult {
          description
            "The if-match rpki origin-as-validation command
             configures a filtering rule based on the BGP
             origin AS validation result.

             Attackers can steal user data by advertising routes
             that are more specific than those advertised by
             carriers. RPKI can address this issue by validating
             the origin ASs of BGP routes and apply the BGP
             origin AS validation result to route selection. The
             validation result can be Valid, Not Found, or
             Invalid.

             To configure a filtering rule based on the BGP
             origin AS validation result, you can run the
             if-match rpki origin-as-validation command. After
             the filtering rule is configured, attribute of
             the routes that match the filtering rule can be
             modified based on the apply clause.";

          config "true";
          mandatory "true";
          type enumeration {
            enum "valid" {
              value "0";
              description "valid:";
            }
            enum "invalid" {
```

```
                value "1";
                description "invalid:";
              }
              enum "notFound" {
                value "2";
                description "notFound:";
              }
            }
          }//End of leaf matchOriginAsValidateResult

          leaf matchExtCmntySooList {
            description
              "The if-match extcommunity-list soo command sets a
               filtering rule that is based on the Source of
               Origin (SoO) extended community filter.

               The extended community attributes help flexibly
               control the route-policy. You can use the if-match
               extcommunity-list soo command to configure a node
               to filter routes based on the SoO extended
               community filter.";

            config "true";
            type "string";
          }

        }//End of container matchCondition


        /*apply clauses*/
        container applyAction {
          description
            "Apply the matching rules to the routing policies for
             route advertisement, reception, and import.";

          container applyAsPaths {
            description
              "The apply as-path command replaces the original
               AS_Path list or add the specified AS number to
               the original AS_Path list when BGP route selection
               is adjusted.";
            container asPathStrings {
              list asPathString {
                key "sequenceNumber";
                min-elements "0";
                max-elements "unbounded";

                leaf sequenceNumber {
```

```
                    config "true";
                    type uint32 {
                      range "1..10";
                    }
                  }
                  leaf stringValue {
                    config "true";
                    mandatory "true";
                    type "string";
                  }
                }
              }
              leaf operationType {
                  config "true";
                  type enumeration {
                    enum "delete" {
                      value "0";
                      description "delete:";
                    }
                    enum "replace" {
                      value "1";
                      description "replace:";
                    }
                    enum "additive" {
                      value "2";
                      description "additive:";
                    }
                    enum "delSpecial" {
                      value "3";
                      description "delSpecial:";
                    }
                  }
                  default "replace";
              }
            }//End of container applyAsPaths

            container applyNextHops {
              description
                "The apply ip-address next-hop command sets the
                 next hop address of a route.
                 In BGP, the next hop address of a route can be
                 set through the route-policy in the following
                 situations:
                 1) IBGP: For an IBGP peer, the configured import
                    and export policies can take effect. If the
                    next hop address configured in the policy is
                    unreachable, the IBGP peer still adds the route
                    to the BGP routing table, but the route is not
```

```
                    valid.
                 2) EBGP: For an EBGP peer, when the policy is used
                    to modify the next hop address of a route, the
                    import policy is configured. If the export policy
                    is configured, the route is discarded because its
                    next hop is unreachable. When the EBGP peer
                    relationship is established through a physical
                    connection, the policy cannot take effect. That
                    is, the next hop address of the route cannot be
                    modified.";

              leaf nextHop {
                config "true";
                mandatory "true";
                  type inet:ipv4-address;
              }
              leaf isPeerAddress {
                config "true";
                type "boolean";
                default "false";
              }
            }//End of container applyNextHops

            leaf applyLocalPreference {
              description
                "The apply local-preference command sets the local
                 preference (Local-Pref) for BGP routes.

                 The Local-Pref attribute is the proprietary
                 attribute of BGP. Therefore, the apply
                 local-preference command sets only the Local-Pref
                 for BGP routes. The Local_Pref attribute is used
                 to determine the optimal route when traffic leaves
                 an AS. When a BGP router obtains multiple routes to
                 the same destination address but with different
                 next hops through IBGP peers, the route with the
                 largest Local_Pref value is selected.";

              config "true";
              mandatory "true";
              type uint32 {
                range "0..4294967295";
              }
            }

            container applyCosts {
              description
                "The apply cost command sets the MED value for BGP
```

```
                routes or cost value for routes of other protocols.
                When the filtering conditions specified by if-match
                clauses are met, you can run the apply cost command
                to change the route MED or cost to control route
                selection. After setting the MED or cost, the MED
                or cost of the routes that are imported using the
                route-policy is changed accordingly.";

              leaf applyChoice {
                config "true";
                mandatory "true";
                type enumeration {
                  enum "Add" {
                    value "0";
                    description "Add:";
                  }
                  enum "Sub" {
                    value "1";
                    description "Sub:";
                  }
                  enum "Replace" {
                    value "2";
                    description "Replace:";
                  }
                  enum "Inherit" {
                    value "3";
                    description "Inherit:";
                  }
                }
              }
              leaf costValue {
                config "true";
                mandatory "true";
                type uint32 {
                  range "0..4294967295";
                }
              }
            }//End of container applyCosts

            container applyIpv6NextHops {
              description
                "The apply ipv6 next-hop command configures an IPv6
                 next hop address for a BGP route through a
                 route-policy.
                 In BGP, the next hop address of a route can be set
                 through the route-policy in the following
                 situations:
                 1) IBGP: For an IBGP peer, the configured inbound
```

```
                    and outbound policies can take effect. If the
                    next hop address configured in the policy is
                    unreachable, the IBGP peer still adds the route
                    to the BGP routing table, but the route is not
                    valid.
                 2) EBGP: For an EBGP peer, when the policy is used
                    to modify the next hop address of a route, the
                    inbound policy is configured. If the outbound
                    policy is configured, the route is discarded
                    because its next hop is unreachable. When the
                    EBGP peer relationship is established through
                    a physical connection, the policy cannot take
                    effect. That is, the next hop address of the
                    route cannot be modified.";

                leaf ipv6NextHop {
                  config "true";
                  mandatory "true";
                  type inet:ipv6-address;
                }
                leaf isPeerAddress {
                  config "true";
                  type "boolean";
                  default "false";
                }
              }//End of container applyIpv6NextHops

              leaf applyCostType {
                description
                  "The apply cost-type command sets the cost type of
                   the route.
                   1) The apply cost-type { external | internal }
                   command sets the cost type of IS-IS routes. The
                   cost of an internal route imported to IS-IS remains
                   unchanged and the cost of an external route imported
                   to IS-IS is increased by 64.
                   2) The apply cost-type { type-1 | type-2 }
                   command modifies the type of OSPF routes. During
                   route import, OSPF modifies the type but not the
                   cost value of the original route. When OSPF
                   advertises the imported route with the cost and
                   type information to a peer, the peer device will
                   recalculate the cost value of the imported
                   route based on the received information.";

                config "true";
                mandatory "true";
                type enumeration {
```

```
                enum "external" {
                  value "0";
                  description "external:";
                }
                enum "internal" {
                  value "1";
                  description "internal:";
                }
                enum "type_1" {
                  value "2";
                  description "type_1:";
                }
                enum "type_2" {
                  value "3";
                  description "type_2:";
                }
              }
            }//End of leaf applyCostType

            container applyOrigin {
              description
                "The apply origin command sets the Origin attribute
                 of BGP routes. The Origin attribute, as a
                 proprietary attribute of BGP, defines the origin of
                 a route. It identifies how a BGP route is generated.
                 You can use the apply origin command to set the
                 Origin attribute of BGP routes.";

              leaf originType {
                config "true";
                mandatory "true";
                type enumeration {
                  enum "egp" {
                    value "0";
                    description "egp:";
                  }
                  enum "igp" {
                    value "1";
                    description "igp:";
                  }
                  enum "incomplete" {
                    value "2";
                    description "incomplete:";
                  }
                }
              }
              leaf asStrOrNum {
                config "true";
```

```
            mandatory "true";
            type "string";
          }
        }//End of container applyOrigin

        container applyCommunitys {
          description
            "The apply community command configures BGP
             community attributes. The community attribute,
             which is the private attribute of BGP, simplifies
             the application of routing policies and facilitates
             route maintenance and management. A community is a
             set of destination addresses with the same
             characteristics. These addresses have no physical
             boundary and are independent of their ASs. They
             share one or multiple community attributes,
             which can be changed or set through the apply
             community command.";

          leaf operationType {
            config "true";
            default "replace";
            type enumeration {
              enum "delete" {
                value "0";
                description "delete:";
              }
              enum "replace" {
                value "1";
                description "replace:";
              }
              enum "additive" {
                value "2";
                description "additive:";
              }
              enum "delSpecial" {
                value "3";
                description "delSpecial:";
              }
            }

          }

          container applyCmntyStrings {
            list applyCmntyString {
              key "stringValue";
              min-elements "0";
              max-elements "unbounded";
```

```
          leaf stringValue {
            config "true";
            type "string";
          }
        }
      }
    }//End of container applyCommunitys

    leaf applyTagValue {
      description
        "The apply tag command sets the tag for routes.
         If routes satisfy the filter condition specified
         by the if-match clause, you can run the apply tag
         command to set the same tag for the routes that
         satisfy the same filter condition to classify the
         routes.";

      config "true";
      mandatory "true";
      type uint32 {
        range "0..4294967295";
      }
    }

    leaf applyMplsLabel {
      description
        "The apply mpls-label command assigns MPLS labels to
         routes. In the scenario where inter-AS VPN Option C
         or carrier's carrier (CSC) is deployed, you can use
         the apply mpls-label command to allocate labels to
         public routes.";

      config "true";
      mandatory "true";
      type "boolean";
    }

    leaf applyPreference {
      description
        "The apply preference command sets the preference
         for routes. If a route satisfies the filter
         condition specified by the if-match clause, you
         can run the apply preference command to change
         the preference of the route to participate in
         route selection. Then, when different protocols
         discover multiple routes to the same destination,
         the route discovered by the protocol with a higher
         preference is selected as a valid route to forward
```

```
                packets.";

              config "true";
              mandatory "true";
              type uint32 {
                range "1..255";
              }
            }

            container applyExtendCommunitys {
              description
                "The apply extcommunity command configures extended
                 community attributes for BGP routes.
                 The apply extcommunity command is applicable to
                 BGP/MPLS IP VPNs. At present, there are two types
                 of BGP extended community attributes.
                   1) VPN route-target (RT) extended community
                   2) Source of Origin (SoO) extended community
                 At present, RT extended community attributes can be
                 set only through the route-policy. This command
                 cannot specify an extended community attribute for
                 public routes.";

              container applyExtCmntyStrings {
                list applyExtCmntyString {
                  key "stringValue";
                  min-elements "0";
                  max-elements "unbounded";

                  leaf stringValue {
                    config "true";
                    type "string";
                  }
                }
              }

              leaf additiveFlag {
                config "true";
                default "false";
                type "boolean";
              }
            }//End of container applyExtendCommunitys

            container applyCommunityFilterDelete {
              description
                "The apply comm-filter delete command deletes a BGP
                 route community according to the specified value in
                 the community filter. The community filter can be
```

```
                    either a basic or advanced community filter.

                    When the apply comm-filter delete command is run in
                    the Route-Policy view to delete the values in the
                    community filter, only one community attribute can
                    be specified in an ip community-filter command. If
                    multiple community attributes are configured in the
                    same community filter, running the apply comm-filter
                    delete command cannot delete these community
                    attributes. To delete the community attributes, you
                    need to run the ip community-filter command several
                    times to configure community attributes one by one,
                    and then run the apply comm-filter delete command to
                    delete these community attributes.";

                  leaf communityNum {
                    config "true";
                    type "string";
                  }
                  leaf communityName {
                    config "true";
                    type "string";
                  }
                }//End of container applyCommunityFilterDelete

                container applyDampening {
                  description
                    "The apply dampening command sets dampening
                     parameters for EBGP routes. The apply dampening
                     command, which is mostly used in BGP, is used
                     to prevent frequent route dampening from
                     affecting routers on the network.
                     You can configure different route dampening
                     parameters for different nodes in the same
                     route-policy. When route flapping occurs, BGP
                     can use different route dampening parameters
                     to suppress the routes that match the
                     route-policy.";

                  leaf halfLifeValue {
                    config "true";
                    mandatory "true";
                    type uint32 {
                      range "1..45";
                    }
                  }
                  leaf reuseValue {
                    config "true";
```

```
            mandatory "true";
            type uint32 {
              range "1..20000";
            }
          }
          leaf suppressValue {
            config "true";
            mandatory "true";
            type uint32 {
              range "1..20000";
            }
          }
          leaf ceilingValue {
            config "true";
            mandatory "true";
            type uint32 {
              range "1001..20000";
            }
          }
        }//End of container applyDampening

        leaf applyRouteType {
          description
            "
            1) After a route matches a route-policy, you can
               set the level of the route imported to IS-IS.
            2) Routes matching the if-match clauses defined
               in the routing policy will be imported into
               the OSPF area specified in the command.";

          config "true";
          mandatory "true";
          type enumeration {
            enum "OspfStubArea" {
              value "0";
              description "OspfStubArea:";
            }
            enum "OspfBackbone" {
              value "1";
              description "OspfBackbone:";
            }
            enum "IsisLevel1" {
              value "2";
              description "IsisLevel1:";
            }
            enum "IsisLevel2" {
              value "3";
              description "IsisLevel2:";
```

```
                }
                enum "IsisLevel12" {
                  value "4";
                  description "IsisLevel12:";
                }
              }
            }//End of leaf applyRouteType

            leaf applyPreferredValue {
              description
                "The apply preferred-value command sets the preferred
                 value for BGP routes.
                 The preferred value is a proprietary attribute of
                 BGP.
                 You can use the apply preferred-value command to set
                 the preferred value for a BGP route in the import
                 policy.";

              config "true";
              mandatory "true";
              type uint32 {
                range "0..65535";
              }
            }//End of leaf applyPreferredValue

            container applyQosParas {
              leaf QosType {
                config "true";
                mandatory "true";
                type enumeration {
                  enum "QosID" {
                    value "0";
                    description "qosLocalID:";
                  }
                  enum "Behavior" {
                    value "1";
                    description "behavior:";
                  }
                  enum "IpPrecedence" {
                    value "2";
                    description "ipPrecedence:";
                  }
                }
              }
              leaf qosLocalID {
                description
                  "The apply qos-local-id command sets the QoS local
                   ID.
```

```
            The QoS local ID is a local identifier of QoS. In
            actual applications, you can set the QoS local ID
            in the route-policy, and add the command that
            matches the QoS local ID in the QoS policy. The QoS
            local ID set in the route-policy is delivered to
            the FIB table. During packet forwarding, the system
            obtains the QoS local ID from the FIB table and
            applies the related QoS policy according to the QoS
            local ID.";

          config "true";
          mandatory "true";
          type uint32 {
            range "1..4095";
          }
        }

        leaf ipPrecedence {
          description
            "The apply ip precedence command sets the QoS
             parameter ip-precedence for routes.

             The apply ip precedence command is generally
             applicable to QPPB, in which the IP precedence
             is a QoS parameter that can be set. After
             receiving routes, a BGP route receiver matches
             the attributes of the BGP routes based on the
             import route-policy, sets the IP precedence,
             delivers the BGP routes together with the
             associated QoS parameters, and applies QoS
             traffic policies to the classified data. In
             this case, the BGP route receiver can apply
             QoS policies to the data sent to the destination
             network segment based on the IP precedence. This
             applies QoS policies in BGP.";

          config "true";
          mandatory "true";
          type uint32 {
            range "0..7";
          }
        }

        leaf behaviorName {
          description
            "The apply behavior command configures a QoS
             traffic behavior for routes.";
```

```
            config "true";
            mandatory "true";
            type "string";
          }
        }//End of container applyQosParas

        leaf applyTrafficIndex {
          description
            "The apply traffic-index command sets the BGP
             traffic index. BGP accounting uses different
             BGP traffic indexes in BGP community
             attributes to identify routes and charge the
             traffic accordingly.";

          config "true";
          mandatory "true";
          type uint32 {
            range "1..64";
          }
        }//End of leaf applyTrafficIndex

        container applyExtCmntySoos {
          description
            "The apply extcommunity soo command configures Source
             of Origin (SoO) extended community attributes for
             BGP routes.";

          leaf operationType {
            config "true";
            type enumeration {
              enum "delete" {
                value "0";
                description "delete:";
              }
              enum "replace" {
                value "1";
                description "replace:";
              }
              enum "additive" {
                value "2";
                description "additive:";
              }
              enum "delSpecial" {
                value "3";
                description "delSpecial:";
              }
            }
            default "replace";
```

```
                }
                container applyExtCmntySooStrings {
                  list applyExtCmntySooString {
                    key "stringValue";
                    min-elements "0";
                    max-elements "unbounded";

                    leaf stringValue {
                      config "true";
                      type "string";
                    }
                  }
                }
              }//End of container applyExtCmntySoos

              leaf applyPriorityValue {
                description
                  "The apply preference command sets the preference
                   for routes.

                   If a route satisfies the filter condition specified
                   by the if-match clause, you can run the apply
                   preference command to change the preference of the
                   route to participate in route selection. Then, when
                   different protocols discover multiple routes to the
                   same destination, the route discovered by the
                   protocol with a higher preference is selected as a
                   valid route to forward packets.";

                config "true";
                mandatory "true";
                type uint16 {
                  range "1..255";
                }
              }//End of leaf applyPriorityValue

            }
          }
        }//End of container routePolicyNodes

      }
    }//End of container routePolicys

  }//End of container routing-policy

}
<CODE ENDS>
```

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

This document does not introduce any new security risk.

8. Acknowledgements

The authors would like to thank Xianping Zhang, Nan Meng, Linghai Kong for their contributions to this work.

9. References

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

[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.

Authors' Addresses

Gang Yan
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: yangang@huawei.com

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com