RTGWG
Internet-Draft
Intended status: Standards Track
Expires: December 3, 2020

Y. Qu
Futurewei
J. Tantsura
Apstra
A. Lindem
Cisco
X. Liu
Volta Networks
June 1, 2020

A YANG Data Model for Routing Policy Management
draft-ietf-rtgwg-policy-model-14

Abstract

This document defines a YANG data model for configuring and managing routing policies in a vendor-neutral way and based on actual operational practice. The model provides a generic policy framework which can be augmented with protocol-specific policy configuration.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 3, 2020.

Copyright Notice

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

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

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

Table of Contents

1. Introduction 2
1.1. Goals and approach 3
2. Terminology and Notation 3
2.1. Tree Diagrams 4
2.2. Prefixes in Data Node Names 4
3. Model overview 5
4. Route policy expression 5
4.1. Defined sets for policy matching 6
4.2. Policy conditions 7
4.3. Policy actions 8
4.4. Policy subroutines 9
5. Policy evaluation 10
6. Applying routing policy 10
7. Routing protocol-specific policies 11
8. Security Considerations 13
9. IANA Considerations 14
10. YANG module 14
10.1. Routing policy model 15
11. Policy examples 36
12. References 37
12.1. Normative references 37
12.2. Informative references 39
Appendix A. Acknowledgements 39
Authors' Addresses 40

1. Introduction

This document describes a YANG [RFC7950] data model for routing policy configuration based on operational usage and best practices in a variety of service provider networks. The model is intended to be vendor-neutral, in order to allow operators to manage policy configuration in a consistent, intuitive way in heterogeneous environments with routers supplied by multiple vendors.

The YANG modules in this document conform to the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Goals and approach

This model does not aim to be feature complete -- it is a subset of the policy configuration parameters available in a variety of vendor implementations, but supports widely used constructs for managing how routes are imported, exported, and modified across different routing protocols. The model development approach has been to examine actual policy configurations in use across a number of operator networks. Hence the focus is on enabling policy configuration capabilities and structure that are in wide use.

Despite the differences in details of policy expressions and conventions in various vendor implementations, the model reflects the observation that a relatively simple condition-action approach can be readily mapped to several existing vendor implementations, and also gives operators an intuitive and straightforward way to express policy without sacrificing flexibility. A side effect of this design decision is that legacy methods for expressing policies are not considered. Such methods could be added as an augmentation to the model if needed.

Consistent with the goal to produce a data model that is vendor neutral, only policy expressions that are deemed to be widely available in existing major implementations are included in the model. Those configuration items that are only available from a single implementation are omitted from the model with the expectation they will be available in separate vendor-provided modules that augment the current model.

2. Terminology and Notation

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

Routing Policy: A routing policy defines how routes are imported, exported, modified, and advertised between routing protocols instances or within a single routing protocol instance.

The following terms are defined in [RFC8342]:

- client
- server
- configuration

- system state
- operational state
- intended configuration

The following terms are defined in [RFC7950]:

- action
- augment
- container
- container with presence
- data model
- data node
- feature
- leaf
- list
- mandatory node
- module
- schema tree
- RPC (Remote Procedure Call) operation

2.1. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

2.2. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
if-ext	ietf-if-extensions	[INTF-EXT-YANG]
if-l3-vlan	ietf-if-l3-vlan	[SUB-INTF-VLAN-YANG]

Table 1: Prefixes and Corresponding YANG Modules

3. Model overview

The routing policy module has three main parts:

- A generic framework to express policies as sets of related conditions and actions. This includes match sets and actions that are useful across many routing protocols.

- A structure that allows routing protocol models to add protocol-specific policy conditions and actions though YANG augmentations. There is a complete example of this for BGP [RFC4271] policies in the proposed vendor-neutral BGP data model [I-D.ietf-idr-bgp-model].

- A reusable grouping for attaching import and export rules in the context of routing configuration for different protocols, VRFs, etc. This also enables creation of policy chains and expressing default policy behavior.

The module makes use of the standard Internet types, such as IP addresses, autonomous system numbers, etc., defined in RFC 6991 [RFC6991].

4. Route policy expression

Policies are expressed as a sequence of top-level policy definitions each of which consists of a sequence of policy statements. Policy statements in turn consist of simple condition-action tuples. Conditions may include multiple match or comparison operations, and similarly, actions may effect multiple changes to route attributes,

or indicate a final disposition of accepting or rejecting the route. This structure is shown below.

```
   +--rw routing-policy
      +--rw policy-definitions
         +--rw policy-definition* [name]
            +--rw name          string
            +--rw statements
               +--rw statement* [name]
                  +--rw name          string
                  +--rw conditions
                  |     ...
                  +--rw actions
                        ...
```

4.1. Defined sets for policy matching

The models provides a set of generic sets that can be used for matching in policy conditions. These sets are applicable for route selection across multiple routing protocols. They may be further augmented by protocol-specific models which have their own defined sets. The supported defined sets include:

- prefix sets - define a set of IP prefixes, each with an associated IP prefix and netmask range (or exact length)

- neighbor sets - define a set of neighboring nodes by their IP addresses. These sets are used for selecting routes based on the neighbors advertising the routes.

- tag set - define a set of generic tag values that can be used in matches for filtering routes

The model structure for defined sets is shown below.

```
      +--rw routing-policy
         +--rw defined-sets
         |  +--rw prefix-sets
         |  |  +--rw prefix-set* [name]
         |  |     +--rw name        string
         |  |     +--rw mode?       enumeration
         |  |     +--rw prefixes
         |  |        +--rw prefix-list* [ip-prefix mask-length-lower
         |  |                            mask-length-upper]
         |  |           +--rw ip-prefix           inet:ip-prefix
         |  |           +--rw mask-length-lower    uint8
         |  |           +--rw mask-length-upper    uint8
         |  +--rw neighbor-sets
         |  |  +--rw neighbor-set* [name]
         |  |     +--rw name       string
         |  |     +--rw address*   inet:ip-address
         |  +--rw tag-sets
         |     +--rw tag-set* [name]
         |        +--rw name         string
         |        +--rw tag-value*   tag-type
```

4.2. Policy conditions

Policy statements consist of a set of conditions and actions (either of which may be empty). Conditions are used to match route attributes against a defined set (e.g., a prefix set), or to compare attributes against a specific value.

Match conditions may be further modified using the match-set-options configuration which allows operators to change the behavior of a match. Three options are supported:

- ALL - match is true only if the given value matches all members of the set.
- ANY - match is true if the given value matches any member of the set.
- INVERT - match is true if the given value does not match any member of the given set.

Not all options are appropriate for matching against all defined sets (e.g., match ALL in a prefix set does not make sense). In the model, a restricted set of match options is used where applicable.

Comparison conditions may similarly use options to change how route attributes should be tested, e.g., for equality or inequality, against a given value.

While most policy conditions will be added by individual routing protocol models via augmentation, this routing policy model includes several generic match conditions and also the ability to test which protocol or mechanism installed a route (e.g., BGP, IGP, static, etc.). The conditions included in the model are shown below.

```
+--rw routing-policy
  +--rw policy-definitions
      +--rw policy-definition* [name]
         +--rw name          string
         +--rw statements
            +--rw statement* [name]
               +--rw conditions
               |  +--rw call-policy?
               |  +--rw source-protocol?
               |  +--rw match-interface
               |  |  +--rw interface?
               |  |  +--rw subinterface?
               |  +--rw match-prefix-set
               |  |  +--rw prefix-set?
               |  |  +--rw match-set-options?
               |  +--rw match-neighbor-set
               |  |  +--rw neighbor-set?
               |  +--rw match-tag-set
               |  |  +--rw tag-set?
               |  |  +--rw match-set-options?
               |  +--rw match-route-type*   identityref
```

4.3. Policy actions

When policy conditions are satisfied, policy actions are used to set various attributes of the route being processed, or to indicate the final disposition of the route, i.e., accept or reject.

Similar to policy conditions, the routing policy model includes generic actions in addition to the basic route disposition actions. These are shown below.

```
      +--rw routing-policy
        +--rw policy-definitions
           +--rw policy-definition* [name]
              +--rw statements
                 +--rw statement* [name]
                   +--rw actions
                       +--rw policy-result?   policy-result-type
                       +--rw set-metric
                       |  +--rw metric-modification?
                       |  |         metric-modification-type
                       |  +--rw metric?                uint32
                       +--rw set-metric-type
                       |  +--rw metric-type?   identityref
                       +--rw set-import-level
                       |  +--rw import-level?   identityref
                       +--rw set-preference?        uint16
                       +--rw set-tag?               tag-type
                       +--rw set-application-tag?   tag-type
```

4.4. Policy subroutines

Policy 'subroutines' (or nested policies) are supported by allowing policy statement conditions to reference other policy definitions using the call-policy configuration. Called policies apply their conditions and actions before returning to the calling policy statement and resuming evaluation. The outcome of the called policy affects the evaluation of the calling policy. If the called policy results in an accept-route, then the subroutine returns an effective boolean true value to the calling policy. For the calling policy, this is equivalent to a condition statement evaluating to a true value and evaluation of the policy continues (see Section 5). Note that the called policy may also modify attributes of the route in its action statements. Similarly, a reject-route action returns false and the calling policy evaluation will be affected accordingly. When the end of the subroutine policy chain is reached, the default route disposition action is returned (i.e., boolean false for reject-route unless an alternate default action is specified for the chain). Consequently, a subroutine cannot explicitly accept or reject a route. Rather it merely provides an indication that 'call-policy' condition returns boolean true or false indicating whether or not the condition matches. Route acceptance or rejection is solely determined by the top-level policy.

Note that the called policy may itself call other policies (subject to implementation limitations). The model does not prescribe a nesting depth because this varies among implementations. For example, some major implementation may only support a single level of subroutine recursion. As with any routing policy construction, care

must be taken with nested policies to ensure that the effective return value results in the intended behavior. Nested policies are a convenience in many routing policy constructions but creating policies nested beyond a small number of levels (e.g., 2-3) should be discouraged.

5. Policy evaluation

Evaluation of each policy definition proceeds by evaluating its corresponding individual policy statements in order. When all the condition statements in a policy statement are satisfied, the corresponding action statements are executed. If the actions include either accept-route or reject-route actions, evaluation of the current policy definition stops, and no further policy definitions in the chain are evaluated.

If the conditions are not satisfied, then evaluation proceeds to the next policy statement. If none of the policy statement conditions are satisfied, then evaluation of the current policy definition stops, and the next policy definition in the chain is evaluated. When the end of the policy chain is reached, the default route disposition action is performed (i.e., reject-route unless an alternate default action is specified for the chain).

Note that the route's pre-policy attributes are always used for testing policy statement conditions. In other words, if actions modify the policy application specific attributes, those modifications are not used for policy statement conditions.

6. Applying routing policy

Routing policy is applied by defining and attaching policy chains in various routing contexts. Policy chains are sequences of policy definitions (described in Section 4) that have an associated direction (import or export) with respect to the routing context in which they are defined. The routing policy model defines an apply-policy grouping that can be imported and used by other models. As shown below, it allows definition of import and export policy chains, as well as specifying the default route disposition to be used when no policy definition in the chain results in a final decision.

```
+--rw apply-policy
|  +--rw import-policy*
|  +--rw default-import-policy?   default-policy-type
|  +--rw export-policy*
|  +--rw default-export-policy?   default-policy-type
```

The default policy defined by the model is to reject the route for both import and export policies.

7. Routing protocol-specific policies

Routing models that require the ability to apply routing policy may augment the routing policy model with protocol or other specific policy configuration. The routing policy model assumes that additional defined sets, conditions, and actions may all be added by other models.

An example of this is shown below, in which the BGP configuration model in [I-D.ietf-idr-bgp-model] adds new defined sets to match on community values or AS paths. The model similarly augments BGP-specific conditions and actions in the corresponding sections of the routing policy model.

```
module: ietf-routing-policy
+--rw routing-policy
  +--rw defined-sets
  |  +--rw prefix-sets
  |  |  +--rw prefix-set* [name]
  |  |     +--rw name        string
  |  |     +--rw mode?       enumeration
  |  |     +--rw prefixes
  |  |        +--rw prefix-list* [ip-prefix mask-length-lower
  |  |                            mask-length-upper]
  |  |           +--rw ip-prefix           inet:ip-prefix
  |  |           +--rw mask-length-lower   uint8
  |  |           +--rw mask-length-upper   uint8
  |  +--rw neighbor-sets
  |  |  +--rw neighbor-set* [name]
  |  |     +--rw name       string
  |  |     +--rw address*   inet:ip-address
  |  +--rw tag-sets
  |  |  +--rw tag-set* [name]
  |  |     +--rw name         string
  |  |     +--rw tag-value*   tag-type
  |  +--rw bp:bgp-defined-sets
  |     +--rw bp:community-sets
  |     |  +--rw bp:community-set* [name]
  |     |     +--rw bp:name      string
  |     |     +--rw bp:member*   union
  |     +--rw bp:ext-community-sets
  |     |  +--rw bp:ext-community-set* [name]
  |     |     +--rw bp:name      string
  |     |     +--rw bp:member*   union
  |     +--rw bp:as-path-sets
```

```
|        +--rw bp:as-path-set* [name]
|           +--rw bp:name      string
|           +--rw bp:member*   string
+--rw policy-definitions
   +--rw policy-definition* [name]
      +--rw name          string
      +--rw statements
         +--rw statement* [name]
            +--rw name          string
            +--rw conditions
            |  +--rw call-policy?
            |  +--rw source-protocol?         identityref
            |  +--rw match-interface
            |  |  +--rw interface?
            |  |  +--rw subinterface?
            |  +--rw match-prefix-set
            |  |  +--rw prefix-set?       prefix-set/name
            |  |  +--rw match-set-options?   match-set-options-type
            |  +--rw match-neighbor-set
            |  |  +--rw neighbor-set?
            |  +--rw match-tag-set
            |  |  +--rw tag-set?
            |  |  +--rw match-set-options?   match-set-options-type
            |  +--rw match-route-type*   identityref
            |  +--rw bp:bgp-conditions
            |     +--rw bp:med-eq?         uint32
            |     +--rw bp:origin-eq?      bt:bgp-origin-attr-type
            |     +--rw bp:next-hop-in*    inet:ip-address-no-zone
            |     +--rw bp:afi-safi-in*    identityref
            |     +--rw bp:local-pref-eq?  uint32
            |     +--rw bp:route-type?     enumeration
            |     +--rw bp:community-count
            |     +--rw bp:as-path-length
            |     +--rw bp:match-community-set
            |     |  +--rw bp:community-set?
            |     |  +--rw bp:match-set-options?
            |     +--rw bp:match-ext-community-set
            |     |  +--rw bp:ext-community-set?
            |     |  +--rw bp:match-set-options?
            |     +--rw bp:match-as-path-set
            |        +--rw bp:as-path-set?
            |        +--rw bp:match-set-options?
            +--rw actions
               +--rw policy-result?         policy-result-type
               +--rw set-metric
               |  +--rw metric-modification?
               |  +--rw metric?               uint32
               +--rw set-metric-type
```

```
|  +--rw metric-type?   identityref
+--rw set-import-level
|  +--rw import-level?   identityref
+--rw set-preference?        uint16
+--rw set-tag?               tag-type
+--rw set-application-tag?   tag-type
+--rw bp:bgp-actions
   +--rw bp:set-route-origin?bt:bgp-origin-attr-type
   +--rw bp:set-local-pref?   uint32
   +--rw bp:set-next-hop?     bgp-next-hop-type
   +--rw bp:set-med?          bgp-set-med-type
   +--rw bp:set-as-path-prepend
   |  +--rw bp:repeat-n?   uint8
   +--rw bp:set-community
   |  +--rw bp:method?      enumeration
   |  +--rw bp:options?
   |  +--rw bp:inline
   |  |  +--rw bp:communities*   union
   |  +--rw bp:reference
   |     +--rw bp:community-set-ref?
   +--rw bp:set-ext-community
      +--rw bp:method?      enumeration
      +--rw bp:options?
      +--rw bp:inline
      |  +--rw bp:communities*   union
      +--rw bp:reference
         +--rw bp:ext-community-set-ref?
```

8. Security Considerations

The YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The NETCONF Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a pre-configured subset of all available NETCONF or RESTCONF protocol operations and content.

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

effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/routing-policy

/routing-policy/defined-sets/prefix-sets

/routing-policy/defined-sets/neighbor-sets

/routing-policy/defined-sets/tag-sets

/routing-policy/policy-definitions

Unauthorized access to any data node of these subtrees can disclose the operational state information of routing policies on this device.

Routing policy configuration has a significant impact on network operations, and, as such, any related model carries potential security risks. Unauthorized access or invalid data could cause major disruption.

9. IANA Considerations

This document registers a URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested to be made:

```
URI: urn:ietf:params:xml:ns:yang:ietf-routing-policy
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

This document registers a YANG module in the YANG Module Names registry [RFC6020].

```
name: ietf-routing-policy
namespace: urn:ietf:params:xml:ns:yang:ietf-routing-policy
prefix: rt-pol
reference: RFC XXXX
```

10. YANG module

The routing policy model is described by the YANG modules in the sections below. [RFC2328], [RFC3101], [RFC5130], and [RFC5302] are referenced here since they are referenced in the YANG model but not elsewhere in the draft.

10.1. Routing policy model

```
<CODE BEGINS> file "ietf-routing-policy@2020-06-01.yang"
module ietf-routing-policy {

  yang-version "1.1";

  namespace "urn:ietf:params:xml:ns:yang:ietf-routing-policy";
  prefix rt-pol;

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface
               Management (NMDA Version)";
  }

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

  import ietf-if-extensions {
    prefix "if-ext";
    reference "RFC YYYY: Common Interface Extension YANG
               Data Models. Please replace YYYY with
               published RFC number for
               draft-ietf-netmod-intf-ext-yang.";
  }

  import ietf-if-l3-vlan {
    prefix "if-l3-vlan";
    reference "RFC ZZZZ: Sub-interface VLAN YANG Data Models.
               Please replace ZZZZ with published RFC number
               for draft-ietf-netmod-sub-intf-vlan-model.";
  }

  organization
```

```
      "IETF RTGWG - Routing Area Working Group";
    contact
      "WG Web:   <http://tools.ietf.org/wg/rtgwg/>
       WG List:  <Email: rtgwg@ietf.org>

       Editor:   Yingzhen Qu
                 <Email: yingzhen.qu@futurewei.com>
                 Jeff Tantsura
                 <Email: jefftant.ietf@gmail.com>
                 Acee Lindem
                 <Email: acee@cisco.com>
                 Xufeng Liu
                 <Email: xufeng.liu.ietf@gmail.com>";

    description
      "This module describes a YANG model for routing policy
       configuration. It is a limited subset of all of the policy
       configuration parameters available in the variety of vendor
       implementations, but supports widely used constructs for
       managing how routes are imported, exported, modified and
       advertised across different routing protocol instances or
       within a single routing protocol instance.  This module is
       intended to be used in conjunction with routing protocol
       configuration modules (e.g., BGP) defined in other models.

       Route policy expression:

       Policies are expressed as a set of top-level policy
       definitions, each of which consists of a sequence of policy
       statements. Policy statements consist of simple
       condition-action tuples. Conditions may include multiple match
       or comparison operations, and similarly actions may be
       multitude of changes to route attributes or a final
       disposition of accepting or rejecting the route.

       Route policy evaluation:

       Policy definitions are referenced in routing protocol
       configurations using import and export configuration
       statements. The arguments are members of an ordered list of
       named policy definitions which comprise a policy chain, and
       optionally, an explicit default policy action (i.e., reject
       or accept).

       Evaluation of each policy definition proceeds by evaluating
       its corresponding individual policy statements in order.  When
       a condition statement in a policy statement is satisfied, the
       corresponding action statement is executed.  If the action
```

statement has either accept-route or reject-route actions, policy evaluation of the current policy definition stops, and no further policy definitions in the chain are evaluated.

If the condition is not satisfied, then evaluation proceeds to the next policy statement. If none of the policy statement conditions are satisfied, then evaluation of the current policy definition stops, and the next policy definition in the chain is evaluated. When the end of the policy chain is reached, the default route disposition action is performed (i.e., reject-route unless an alternate default action is specified for the chain).

Policy 'subroutines' (or nested policies) are supported by allowing policy statement conditions to reference another policy definition which applies conditions and actions from the referenced policy before returning to the calling policy statement and resuming evaluation. If the called policy results in an accept-route (either explicit or by default), then the subroutine returns an effective true value to the calling policy. Similarly, a reject-route action returns false. If the subroutine returns true, the calling policy continues to evaluate the remaining conditions (using a modified route if the subroutine performed any changes to the route).

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

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

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

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

This version of this YANG module is part of RFC XXXX;
see the RFC itself for full legal notices.";

```
    revision "2020-06-01" {
      description
        "Initial revision.";
      reference
       "RFC XXXX: Routing Policy Configuration Model for Service
        Provider Networks";
    }

    /* Identities */

    identity metric-type {
      description
        "Base identity for route metric types.";
    }

    identity ospf-type-1-metric {
      base metric-type;
      description
        "Identity for the OSPF type 1 external metric types. It
         is only applicable to OSPF routes.";
      reference
        "RFC 2328 - OSPF Version 2";
    }

    identity ospf-type-2-metric {
      base metric-type;
      description
        "Identity for the OSPF type 2 external metric types. It
         is only applicable to OSPF routes.";
      reference
        "RFC 2328 - OSPF Version 2";
    }

    identity isis-internal-metric {
      base metric-type;
      description
        "Identity for the IS-IS internal metric types. It is only
         applicable to IS-IS routes.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }

    identity isis-external-metric {
      base metric-type;
      description
        "Identity for the IS-IS external metric types. It is only
         applicable to IS-IS routes.";
```

```
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }

    identity import-level {
      description
        "Base identity for route import level.";
    }

    identity ospf-normal {
      base import-level;
      description
        "Identity for OSPF importation into normal areas
         It is only applicable to routes imported
         into the OSPF protocol.";
      reference
        "RFC 2328 - OSPF Version 2";
    }

    identity ospf-nssa-only {
      base import-level;
      description
        "Identity for the OSPF Not-So-Stubby Area (NSSA) area
         importation. It is only applicable to routes imported
         into the OSPF protocol.";
      reference
        "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
    }

    identity ospf-normal-nssa {
      base import-level;
      description
        "Identity for OSPF importation into both normal and NSSA
         areas, it is only applicable to routes imported into
         the OSPF protocol.";
      reference
        "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
    }

    identity isis-level-1 {
      base import-level;
      description
        "Identity for IS-IS Level 1 area importation. It is only
         applicable to routes imported into the IS-IS protocol.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
```

```
    }

    identity isis-level-2 {
      base import-level;
      description
        "Identity for IS-IS Level 2 area importation. It is only
         applicable to routes imported into the IS-IS protocol.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }

    identity isis-level-1-2 {
      base import-level;
      description
        "Identity for IS-IS Level 1 and Level 2 area importation. It
         is only applicable to routes imported into the IS-IS
         protocol.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }

    identity proto-route-type {
      description
        "Base identity for route type within a protocol.";
    }

    identity isis-level-1-type {
      base proto-route-type;
      description
        "Identity for IS-IS Level 1 route type. It is only
         applicable to IS-IS routes.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }

    identity isis-level-2-type {
      base proto-route-type;
      description
        "Identity for IS-IS Level 2 route type. It is only
         applicable to IS-IS routes.";
      reference
        "RFC 5302 - Domain-Wide Prefix Distributino with
         Two-Level IS-IS";
    }
```

```
  identity ospf-internal-type {
    base proto-route-type;
    description
      "Identity for OSPF intra-area or inter-area route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328 - OSPF Version 2";
  }

  identity ospf-external-type {
    base proto-route-type;
    description
      "Identity for OSPF external type 1/2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328 - OSPF Version 2";
  }

  identity ospf-external-t1 {
    base ospf-external-type;
    description
      "Identity for OSPF external type 1 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328 - OSPF Version 2";
  }

  identity ospf-external-t2-type {
    base ospf-external-type;
    description
      "Identity for OSPF external type 2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 2328 - OSPF Version 2";
  }

  identity ospf-nssa-type {
    base proto-route-type;
    description
      "Identity for OSPF NSSA type 1/2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity ospf-nssa-t1 {
    base ospf-nssa-type;
    description
```

```
      "Identity for OSPF NSSA type 1 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity ospf-nssa-t2 {
    base ospf-nssa-type;
    description
      "Identity for OSPF NSSA type 2 route type.
       It is only applicable to OSPF routes.";
    reference
      "RFC 3101: The OSPF Not-So-Stubby Area (NSSA) Option";
  }

  identity bgp-local {
    base proto-route-type;
    description
      "Identity for BGP local route type.
       It is only applicable to BGP routes.";
    reference
      "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
  }

  identity bgp-external {
    base proto-route-type;
    description
      "Identity for BGP external route type.
       It is only applicable to BGP routes.";
    reference
      "RFC 4271: A Border Gateway Protocol 4 (BGP-4)";
  }

  /* Type Definitions */

  typedef default-policy-type {
    type enumeration {
      enum accept-route {
        description
          "Default policy to accept the route.";
      }
      enum reject-route {
        description
          "Default policy to reject the route.";
      }
    }
    description
      "Type used to specify route disposition in
```

```
          a policy chain. This typedef retained for
          name compatibility with default import and
          export policy.";
      }

      typedef policy-result-type {
        type enumeration {
          enum accept-route {
            description
              "Policy accepts the route.";
          }
          enum reject-route {
            description
              "Policy rejects the route.";
          }
        }
        description
          "Type used to specify route disposition in
           a policy chain.";
      }

      typedef tag-type {
        type union {
          type uint32;
          type yang:hex-string;
        }
        description
          "Type for expressing route tags on a local system,
           including IS-IS and OSPF; may be expressed as either decimal
           or hexadecimal integer.";
        reference
          "RFC 2328 - OSPF Version 2
           RFC 5130 - A Policy Control Mechanism in IS-IS Using
                      Administrative Tags";
      }

      typedef match-set-options-type {
        type enumeration {
          enum any {
            description
              "Match is true if given value matches any member
               of the defined set.";
          }
          enum all {
            description
              "Match is true if given value matches all
               members of the defined set.";
          }
```

```
        enum invert {
          description
            "Match is true if given value does not match any
             member of the defined set.";
        }
      }
      default any;
      description
        "Options that govern the behavior of a match statement.  The
         default behavior is any, i.e., the given value matches any
         of the members of the defined set.";
    }

    typedef metric-modification-type {
      type enumeration {
        enum set-metric {
          description
            "Set the metric to the specified value.";
        }
        enum  add-metric {
          description
            "Add the specified value to the existing metric.
             If the result would overflow the maximum metric
             (0xffffffff), set the metric to the maximum.";
        }
        enum  subtract-metric {
          description
            "Subtract the specified value to the existing metric. If
             the result would be less than 0, set the metric to 0.";
        }
      }
      description
        "Type used to specify how to set the metric given the
         specified value.";
    }

    /* Groupings */

    grouping prefix {
      description
        "Configuration data for a prefix definition.";

      leaf ip-prefix {
        type inet:ip-prefix;
        mandatory true;
        description
          "The IP prefix represented as an IPv6 or IPv4 network
           number followed prefix length with an intervening slash
```

```
            character a delimiter --  while the prefix may be either
            IPv4 or IPv6, most implementations require all members
            of the prefix set to be the same address family.  Mixing
            address types in the same prefix set is likely to cause
            an error.";
        }

        leaf mask-length-lower {
          type uint8;
          description
            "Mask length range lower bound.";
        }
        leaf mask-length-upper {
          type uint8 {
            range "1..128";
          }
          must "../mask-length-upper >= ../mask-length-lower" {
            error-message "The upper bound should not be less"
                        + "than lower bound.";
          }
          description
            "Mask length range upper bound.

            The combination of mask-length-lower and mask-length-upper
            define a range for the mask length, or single 'exact'
            length if mask-length-lower and mask-length-upper are
            equal.

            Example: 192.0.2.0/24 through 192.0.2.0/26 would be
            expressed as prefix: 192.0.2.0/24,
                         mask-length-lower=24,
                         mask-length-upper=26

            Example: 192.0.2.0/24 (an exact match) would be
            expressed as prefix: 192.0.2.0/24,
                         mask-length-lower=24,
                         mask-length-upper=24";
        }
      }

      grouping match-set-options-group {
        description
          "Grouping containing options relating to how a particular set
           should be matched.";

        leaf match-set-options {
          type match-set-options-type;
          description
```

```
          "Optional parameter that governs the behavior of the
           match operation.";
      }
    }

    grouping match-set-options-restricted-group {
      description
        "Grouping for a restricted set of match operation
         modifiers.";

      leaf match-set-options {
        type match-set-options-type {
          enum any {
            description
              "Match is true if given value matches any
               member of the defined set.";
          }
          enum invert {
            description
              "Match is true if given value does not match
               any member of the defined set.";
          }
        }
        description
          "Optional parameter that governs the behavior of the
           match operation. This leaf only supports matching on
           'any' member of the set or 'invert' the match.
           Matching on 'all' is not supported.";
      }
    }

    grouping match-interface-condition {
      description
        "This grouping provides interface match condition.";

      container match-interface {
        leaf interface {
          type leafref {
            path "/if:interfaces/if:interface/if:name";
          }
          description
            "Reference to a base interface.  If a reference to a
             subinterface is required, this leaf must be specified
             to indicate the base interface.";
        }
        leaf subinterface {
          type leafref {
            path "/if:interfaces/if:interface/if-ext:encapsulation"
```

```
              + "/if-l3-vlan:dot1q-vlan"
              + "/if-l3-vlan:outer-tag/if-l3-vlan:vlan-id";
          }
          description
            "Reference to a subinterface -- this requires the base
             interface to be specified using the interface leaf in
             this container.  If only a reference to a base interface
             is required, this leaf should not be set.";
        }

        description
          "Container for interface match conditions";
      }
    }

    grouping match-route-type-condition {
      description
        "This grouping provides route-type match condition";

      leaf-list match-route-type {
          type identityref {
            base proto-route-type;
          }
          description
            "Condition to check the protocol specific type
             of route. This is normally used during route
             importation to select routes or to set protocol
             specific attributes based on the route type.";
      }
    }

    grouping prefix-set-condition {
      description
        "This grouping provides prefix-set conditions.";

      container match-prefix-set {
        leaf prefix-set {
          type leafref {
            path "../../../../../../../defined-sets/" +
              "prefix-sets/prefix-set/name";
          }
          description
            "References a defined prefix set.";
        }
        uses match-set-options-restricted-group;

        description
          "Match a referenced prefix-set according to the logic
```

```
            defined in the match-set-options leaf.";
      }
    }

    grouping neighbor-set-condition {
      description
        "This grouping provides neighbor-set conditions.";

      container match-neighbor-set {
        leaf neighbor-set {
          type leafref {
            path "../../../../../../../defined-sets/neighbor-sets/" +
            "neighbor-set/name";
            require-instance true;
          }
          description
            "References a defined neighbor set.";
        }

        description
          "Match a referenced neighbor set according to the logic
           defined in the match-set-options-leaf.";
      }
    }

    grouping tag-set-condition {
      description
        "This grouping provides tag-set conditions.";

      container match-tag-set {
        leaf tag-set {
          type leafref {
            path  "../../../../../../../defined-sets/tag-sets" +
            "/tag-set/name";
            require-instance true;
          }
          description
            "References a defined tag set.";
        }
        uses match-set-options-restricted-group;

        description
          "Match a referenced tag set according to the logic defined
           in the match-options-set leaf.";
      }
    }

    grouping apply-policy-import {
```

```
      description
        "Grouping for applying import policies.";

      leaf-list import-policy {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
            "rt-pol:policy-definition/rt-pol:name";
          require-instance true;
        }
        ordered-by user;
        description
          "List of policy names in sequence to be applied on
           receiving a routing update in the current context, e.g.,
           for the current peer group, neighbor, address family,
           etc.";
      }

      leaf default-import-policy {
        type default-policy-type;
        default reject-route;
        description
          "Explicitly set a default policy if no policy definition
           in the import policy chain is satisfied.";
      }

    }

    grouping apply-policy-export {
      description
        "Grouping for applying export policies.";

      leaf-list export-policy {
        type leafref {
          path "/rt-pol:routing-policy/rt-pol:policy-definitions/" +
            "rt-pol:policy-definition/rt-pol:name";
          require-instance true;
        }
        ordered-by user;
        description
          "List of policy names in sequence to be applied on
           sending a routing update in the current context, e.g.,
           for the current peer group, neighbor, address family,
           etc.";
      }

      leaf default-export-policy {
        type default-policy-type;
        default reject-route;
```

```
        description
          "Explicitly set a default policy if no policy definition
           in the export policy chain is satisfied.";
      }
    }

    grouping apply-policy-group {
      description
        "Top level container for routing policy applications. This
         grouping is intended to be used in routing models where
         needed.";

      container apply-policy {
        description
          "Anchor point for routing policies in the model.
           Import and export policies are with respect to the local
           routing table, i.e., export (send) and import (receive),
           depending on the context.";

        uses apply-policy-import;
        uses apply-policy-export;

      }
    }

    container routing-policy {
      description
        "Top-level container for all routing policy.";

      container defined-sets {
        description
          "Predefined sets of attributes used in policy match
           statements.";

        container prefix-sets {
          description
            "Data definitions for a list of IPv4 or IPv6
            prefixes which are matched as part of a policy.";
          list prefix-set {
            key "name";
            description
              "List of the defined prefix sets";

            leaf name {
              type string;
              description
                "Name of the prefix set -- this is used as a label to
                 reference the set in match conditions.";
```

```
        }

        leaf mode {
          type enumeration {
            enum ipv4 {
              description
                "Prefix set contains IPv4 prefixes only.";
            }
            enum ipv6 {
              description
                "Prefix set contains IPv6 prefixes only.";
            }
            enum mixed {
              description
                "Prefix set contains mixed IPv4 and IPv6
                 prefixes.";
            }
          }
          description
            "Indicates the mode of the prefix set, in terms of
             which address families (IPv4, IPv6, or both) are
             present. The mode provides a hint, but the device
             must validate that all prefixes are of the indicated
             type, and is expected to reject the configuration if
             there is a discrepancy.  The MIXED mode may not be
             supported on devices that require prefix sets to be
             of only one address family.";
        }

        container prefixes {
          description
            "Container for the list of prefixes in a policy
             prefix list.";

          list prefix-list {
            key "ip-prefix mask-length-lower mask-length-upper";
            description
              "List of prefixes in the prefix set.";

            uses prefix;
          }
        }
      }
    }

    container neighbor-sets {
      description
        "Data definition for a list of IPv4 or IPv6
```

```
            neighbors which can be matched in a routing policy.";

          list neighbor-set {
            key "name";
            description
              "List of defined neighbor sets for use in policies.";

            leaf name {
              type string;
              description
                "Name of the neighbor set -- this is used as a label
                 to reference the set in match conditions.";
            }

            leaf-list address {
              type inet:ip-address;
              description
                "List of IP addresses in the neighbor set.";
            }
          }
        }

        container tag-sets {
          description
            "Data definitions for a list of tags which can
             be matched in policies.";

          list tag-set {
            key "name";
            description
              "List of tag set definitions.";

            leaf name {
              type string;
              description
                "Name of the tag set -- this is used as a label to
                 reference the set in match conditions.";
            }

            leaf-list tag-value {
              type tag-type;
              description
                "Value of the tag set member.";
            }
          }
        }
      }
```

```
       container policy-definitions {
         description
           "Enclosing container for the list of top-level policy
            definitions.";

         list policy-definition {
           key "name";
           description
             "List of top-level policy definitions, keyed by unique
              name.  These policy definitions are expected to be
              referenced (by name) in policy chains specified in
              import or export configuration statements.";

           leaf name {
             type string;
             description
               "Name of the top-level policy definition -- this name
                is used in references to the current policy.";
           }

           container statements {
             description
               "Enclosing container for policy statements.";

             list statement {
               key "name";
               ordered-by user;
               description
                 "Policy statements group conditions and actions
                  within a policy definition.  They are evaluated in
                  the order specified (see the description of policy
                  evaluation at the top of this module.";

               leaf name {
                 type string;
                 description
                   "Name of the policy statement.";
               }

               container conditions {
                 description
                   "Condition statements for the current policy
                    statement.";

                 leaf call-policy {
                   type leafref {
                     path "../../../../../../" +
                          "rt-pol:policy-definitions/" +
```

```
                "rt-pol:policy-definition/rt-pol:name";
            require-instance true;
          }
          description
            "Applies the statements from the specified policy
             definition and then returns control the current
             policy statement. Note that the called policy
             may itself call other policies (subject to
             implementation limitations). This is intended to
             provide a policy 'subroutine' capability.  The
             called policy should contain an explicit or a
             default route disposition that returns an
             effective true (accept-route) or false
             (reject-route), otherwise the behavior may be
             ambiguous and implementation dependent.";
        }

        leaf source-protocol {
          type identityref {
            base rt:control-plane-protocol;
          }
          description
            "Condition to check the protocol / method used to
            install the route into the local routing table.";
        }

        uses match-interface-condition;
        uses prefix-set-condition;
        uses neighbor-set-condition;
        uses tag-set-condition;
        uses match-route-type-condition;
      }

      container actions {
        description
          "Top-level container for policy action
           statements.";
        leaf policy-result {
          type policy-result-type;
          description
            "Select the final disposition for the route,
             either accept or reject.";
        }
        container set-metric {
          leaf metric-modification {
            type metric-modification-type;
            description
              "Indicates how to modify the metric.";
```

```
      }
      leaf metric {
        type uint32;
        description
          "Metric value to set, add, or subtract.";
      }
      description
        "Set the metric for the route.";
    }
    container set-metric-type {
      leaf metric-type {
        type identityref {
          base metric-type;
        }
        description
          "Route metric type.";
      }
      description
        "Set the metric type for the route.";
    }
    container set-import-level {
      leaf import-level {
        type identityref {
          base import-level;
        }
        description
          "Route importation level.";
      }
      description
        "Set the import level for importation of
         routes.";
    }
    leaf set-preference {
      type uint16;
      description
        "Set the preference for the route.";
    }
    leaf set-tag {
      type tag-type;
      description
        "Set the tag for the route.";
    }
    leaf set-application-tag {
      type tag-type;
      description
        "Set the application tag for the route.";
    }
  }
```

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

11. Policy examples

Below we show an example of XML-encoded configuration data using the routing policy and BGP models to illustrate both how policies are defined, and also how they can be applied. Note that the XML has been simplified for readability.

```
  <config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <routing-policy
     xmlns="urn:ietf:params:xml:ns:yang:ietf-routing-policy">

        <defined-sets>
          <prefix-sets>
            <prefix-set>
              <name>prefix-set-A</name>
              <prefixes>
                <prefix-list>
                  <ip-prefix>192.0.2.0/24</ip-prefix>
                  <mask-length-lower>24</mask-length-lower>
                  <mask-length-upper>32</mask-length-upper>
                </prefix-list>
                <prefix-list>
                  <ip-prefix>10.0.0.0/16</ip-prefix>
                  <mask-length-lower>16</mask-length-lower>
                  <mask-length-upper>32</mask-length-upper>
                </prefix-list>
              </prefixes>
            </prefix-set>
          </prefix-sets>
          <tag-sets>
            <tag-set>
              <name>cust-tag1</name>
              <tag-value>10</tag-value>
            </tag-set>
          </tag-sets>
        </defined-sets>

        <policy-definitions>
         <policy-definition>
```

```
            <name>export-tagged-BGP</name>
            <statements>
              <statement>
                <name>term-0</name>
                <conditions>
                  <match-tag-set>
                    <tag-set>cust-tag1</tag-set>
                  </match-tag-set>
                </conditions>
                <actions>
                  <policy-result>accept-route</policy-result>
                </actions>
              </statement>
            </statements>
          </policy-definition>
        </policy-definitions>

        </routing-policy>
    </config>
```

12. References

12.1. Normative references

[INTF-EXT-YANG]
Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj,, "Common Interface Extension YANG Data Models", 2019, <https://datatracker.ietf.org/doc/draft-ietf-netmod-intf-ext-yang/>.

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

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3101] Murphy, P., "The OSPF Not-So-Stubby Area (NSSA) Option", RFC 3101, DOI 10.17487/RFC3101, January 2003, <https://www.rfc-editor.org/info/rfc3101>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

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

[RFC5130] Previdi, S., Shand, M., Ed., and C. Martin, "A Policy Control Mechanism in IS-IS Using Administrative Tags", RFC 5130, DOI 10.17487/RFC5130, February 2008, <https://www.rfc-editor.org/info/rfc5130>.

[RFC5302] Li, T., Smit, H., and T. Przygienda, "Domain-Wide Prefix Distribution with Two-Level IS-IS", RFC 5302, DOI 10.17487/RFC5302, October 2008, <https://www.rfc-editor.org/info/rfc5302>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

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

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, DOI 10.17487/RFC8349, March 2018, <https://www.rfc-editor.org/info/rfc8349>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[SUB-INTF-VLAN-YANG] Wilton, R., Ball, D., tapsingh@cisco.com, t., and S. Sivaraj, "Sub-interface VLAN YANG Data Model", 2019, <https://datatracker.ietf.org/doc/draft-ietf-netmod-sub-intf-vlan-model/>.

12.2. Informative references

[I-D.ietf-idr-bgp-model] Jethanandani, M., Patel, K., Hares, S., and J. Haas, "BGP YANG Model for Service Provider Networks", draft-ietf-idr-bgp-model-08 (work in progress), February 2020.

Appendix A. Acknowledgements

The routing policy module defined in this draft is based on the OpenConfig route policy model. The authors would like to thank to OpenConfig for their contributions, especially Anees Shaikh, Rob Shakir, Kevin D'Souza, and Chris Chase.

The authors are grateful for valuable contributions to this document and the associated models from: Ebben Aires, Luyuan Fang, Josh George, Stephane Litkowski, Ina Minei, Carl Moberg, Eric Osborne, Steve Padgett, Juergen Schoenwaelder, Jim Uttaro, Russ White, and John Heasley.

Thanks to Mahesh Jethanandani and Tom Petch for their review and comments.

Authors' Addresses

Yingzhen Qu
Futurewei
2330 Central Expressway
Santa Clara CA 95050
USA

Email: yingzhen.qu@futurewei.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Acee Lindem
Cisco
301 Midenhall Way
Cary, NC 27513
US

Email: acee@cisco.com

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com