Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 23, 2018

X. Liu
Jabil
Y. Qu
Futurewei Technologies, Inc.
A. Lindem
Cisco Systems
C. Hopps
Deutsche Telekom
L. Berger
LabN Consulting, L.L.C.

September 19, 2017

Routing Area Common YANG Data Types
draft-ietf-rtgwg-routing-types-14

Abstract

This document defines a collection of common data types using the YANG data modeling language. These derived common types are designed to be imported by other modules defined in the routing area.

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 March 23, 2018.

Copyright Notice

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

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

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

Table of Contents

1. Introduction 2
1.1. Terminology 2
2. Overview 2
3. IETF Routing Types YANG Module 6
4. IANA Routing Types YANG Module 22
5. IANA Considerations 31
5.1. IANA-Maintained iana-routing-types Module 32
6. Security Considerations 33
7. Acknowledgements 33
8. References 34
8.1. Normative References 34
8.2. Informative References 34
Authors' Addresses 36

1. Introduction

The YANG [RFC6020] [RFC7950] is a data modeling language used to model configuration data, state data, Remote Procedure Calls, and notifications for network management protocols. The YANG language supports a small set of built-in data types and provides mechanisms to derive other types from the built-in types.

This document introduces a collection of common data types derived from the built-in YANG data types. The derived types are designed to be the common types applicable for modeling in the routing area.

1.1. Terminology

The terminology for describing YANG data models is found in [RFC7950].

2. Overview

This document defines the two models for common routing types, ietf-routing-types and iana-routing-types. The only module imports are from [RFC6991]. The ietf-routing-types model contains common routing types other than those corresponding directly to IANA mappings. These include:

router-id

Router Identifiers are commonly used to identify nodes in routing and other control plane protocols. An example usage of router-id can be found in [I-D.ietf-ospf-yang].

route-target
Route Targets (RTs) are commonly used to control the distribution of virtual routing and forwarding (VRF) information, see [RFC4364], in support of BGP/MPLS IP virtual private networks (VPNs) and BGP/MPLS Ethernet VPNs [RFC7432]. An example usage can be found in [I-D.ietf-bess-l2vpn-yang].

ipv6-route-target
IPv6 Route Targets (RTs) are similar to standard Route Targets only they are IPv6 Address Specific BGP Extended Communities as described in [RFC5701]. An IPv6 Route Target is 20 octets and includes an IPv6 address as the global administrator.

route-target-type
This type defines the import and export rules of Route Targets, as described in Section 4.3.1 of [RFC4364]. An example usage can be found in [I-D.ietf-idr-bgp-model].

route-distinguisher
Route Distinguishers (RDs) are commonly used to identify separate routes in support of virtual private networks (VPNs). For example, in [RFC4364], RDs are commonly used to identify independent VPNs and VRFs, and more generally, to identify multiple routes to the same prefix. An example usage can be found in [I-D.ietf-idr-bgp-model].

route-origin
Route Origin is commonly used to indicate the Site of Origin for Routing and forwarding (VRF) information, see [RFC4364], in support of BGP/MPLS IP virtual private networks (VPNs) and BGP/MPLS Ethernet VPNs [RFC7432]. An example usage can be found in [I-D.ietf-bess-l3vpn-yang].

ipv6-route-origin
An IPv6 Route Origin would also be used to indicate the Site of Origin for Routing and forwarding (VRF) information, see [RFC4364], in support of virtual private networks (VPNs). IPv6 Route Origins are IPv6 Address Specific BGP Extended Communities as described in [RFC5701]. An IPv6 Route Origin is 20 octets and includes an IPv6 address as the global administrator.

ipv4-multicast-group-address

This type defines the representation of an IPv4 multicast group address, which is in the range from 224.0.0.0 to 239.255.255.255. An example usage can be found in [I-D.ietf-pim-yang].

ipv6-multicast-group-address
This type defines the representation of an IPv6 multicast group address, which is in the range of FF00::/8. An example usage can be found in [I-D.ietf-pim-yang].

ip-multicast-group-address
This type represents an IP multicast group address and is IP version neutral. The format of the textual representation implies the IP version. An example usage can be found in [I-D.ietf-pim-yang].

ipv4-multicast-source-address
IPv4 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e., "*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

ipv6-multicast-source-address
IPv6 source address type for use in multicast control protocols. This type also allows the indication of wildcard sources, i.e., "*". An example of where this type may/will be used is [I-D.ietf-pim-yang].

bandwidth-ieee-float32
Bandwidth in IEEE 754 floating point 32-bit binary format [IEEE754]. Commonly used in Traffic Engineering control plane protocols. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

link-access-type
This type identifies the IGP link type. An example of where this type may/will be used is [I-D.ietf-ospf-yang].

timer-multiplier
This type is used in conjunction with a timer-value type. It is generally used to indicate define the number of timer-value intervals that may expire before a specific event must occur. Examples of this include the arrival of any BFD packets, see [RFC5880] Section 6.8.4, or hello_interval in [RFC3209]. Example of where this type may/will be used is [I-D.ietf-idr-bgp-model] and [I-D.ietf-teas-yang-rsvp].

timer-value-seconds16

This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint16 (2 octets). An example of where this type may/will be used is [I-D.ietf-ospf-yang].

timer-value-seconds32
This type covers timers which can be set in seconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). An example of where this type may/will be used is [I-D.ietf-teas-yang-rsvp].

timer-value-milliseconds
This type covers timers which can be set in milliseconds, not set, or set to infinity. This type supports a range of values that can be represented in a uint32 (4 octets). Examples of where this type may/will be used include [I-D.ietf-teas-yang-rsvp] and [I-D.ietf-bfd-yang].

percentage
This type defines a percentage with a range of 0-100%. An example usage can be found in [I-D.ietf-idr-bgp-model].

timeticks64
This type is based on the timeticks type defined in [RFC6991] but with 64-bit precision. It represents the time in hundredths of a second between two epochs. An example usage can be found in [I-D.ietf-idr-bgp-model].

uint24
This type defines a 24-bit unsigned integer. It is used by target="I-D.ietf-ospf-yang"/>.

generalized-label
This type represents a generalized label for Generalized Multi-Protocol Label Switching (GMPLS) [RFC3471]. The Generalized Label does not identify its type, which is known from the context. An example usage can be found in [I-D.ietf-teas-yang-te].

mpls-label-special-purpose
This type represents the special-purpose Multiprotocol Label Switching (MPLS) label values [RFC7274]. An example usage can be found in [I-D.ietf-mpls-base-yang].

mpls-label-general-use
The 20 bits label values in an MPLS label stack entry, specified in [RFC3032]. This label value does not include the encodings of Traffic Class and TTL (time to live). The label range specified by this type is for general use, with special-purpose MPLS label

values excluded. An example usage can be found in [I-D.ietf-mpls-base-yang].

mpls-label
The 20 bits label values in an MPLS label stack entry, specified in [RFC3032]. This label value does not include the encodings of Traffic Class and TTL (time to live). The label range specified by this type covers the general use values and the special-purpose label values. An example usage can be found in [I-D.ietf-mpls-base-yang].

This document defines the following YANG groupings:

mpls-label-stack
This grouping defines a reusable collection of schema nodes representing an MPLS label stack [RFC3032]. An example usage can be found in [I-D.ietf-mpls-base-yang].

vpn-route-targets
This grouping defines a reusable collection of schema nodes representing Route Target import-export rules used in the BGP enabled Virtual Private Networks (VPNs). [RFC4364][RFC4664]. An example usage can be found in [I-D.ietf-bess-l2vpn-yang].

The iana-routing-types model contains common routing types corresponding directly to IANA mappings. These include:

address-family
This type defines values for use in address family identifiers. The values are based on the IANA Address Family Numbers Registry [IANA-ADDRESS-FAMILY-REGISTRY]. An example usage can be found in [I-D.ietf-idr-bgp-model].

subsequent-address-family
This type defines values for use in subsequent address family (SAFI) identifiers. The values are based on the IANA Subsequent Address Family Identifiers (SAFI) Parameters Registry [IANA-SAFI-REGISTRY].

3. IETF Routing Types YANG Module

```
<CODE BEGINS> file "ietf-routing-types@2017-09-19.yang"
module ietf-routing-types {
  namespace "urn:ietf:params:xml:ns:yang:ietf-routing-types";
  prefix rt-types;

  import ietf-yang-types {
    prefix yang;
```

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

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

        Editor:   Xufeng Liu
                  <mailto:Xufeng_Liu@jabail.com>
                  Yingzhen Qu
                  <mailto:yingzhen.qu@huawei.com>
                  Acee Lindem
                  <mailto:acee@cisco.com>
                  Christian Hopps
                  <mailto:chopps@chopps.org>
                  Lou Berger
                  <mailto:lberger@labn.com>";
     description
       "This module contains a collection of YANG data types
        considered generally useful for routing protocols.

        Copyright (c) 2017 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
        (http://trustee.ietf.org/license-info).

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

     revision 2017-09-19 {
       description "Initial revision.";
       reference "RFC TBD: Routing YANG Data Types";
     }

     /*** Identities related to MPLS/GMPLS ***/

     identity mpls-label-special-purpose-value {
       description
```

```
         "Base identity for deriving identities describing
          special-purpose Multiprotocol Label Switching (MPLS) label
          values.";
       reference
         "RFC7274: Allocating and Retiring Special-Purpose MPLS
          Labels.";
     }

     identity ipv4-explicit-null-label {
       base mpls-label-special-purpose-value;
       description
         "This identity represents the IPv4 Explicit NULL Label.";
       reference "RFC3032: MPLS Label Stack Encoding. Section 2.1.";
     }

     identity router-alert-label {
       base mpls-label-special-purpose-value;
       description
         "This identity represents the Router Alert Label.";
       reference "RFC3032: MPLS Label Stack Encoding. Section 2.1.";
     }

     identity ipv6-explicit-null-label {
       base mpls-label-special-purpose-value;
       description
         "This identity represents the IPv6 Explicit NULL Label.";
       reference "RFC3032: MPLS Label Stack Encoding. Section 2.1.";
     }

     identity implicit-null-label {
       base mpls-label-special-purpose-value;
       description
         "This identity represents the Implicit NULL Label.";
       reference "RFC3032: MPLS Label Stack Encoding. Section 2.1.";
     }

     identity entropy-label-indicator {
       base mpls-label-special-purpose-value;
       description
         "This identity represents the Entropy Label Indicator.";
       reference
         "RFC6790: The Use of Entropy Labels in MPLS Forwarding.
          Sections 3 and 10.1.";
     }

     identity gal-label {
       base mpls-label-special-purpose-value;
       description
```

```
        "This identity represents the Generic Associated Channel
         Label (GAL).";
      reference
        "RFC5586: MPLS Generic Associated Channel.
         Sections 4 and 10.";
    }

    identity oam-alert-label {
      base mpls-label-special-purpose-value;
      description
        "This identity represents the OAM Alert Label.";
      reference
        "RFC3429: Assignment of the 'OAM Alert Label' for
         Multiprotocol Label Switching Architecture (MPLS)
         Operation and Maintenance (OAM) Functions.
         Sections 3 and 6.";
    }

    identity extension-label {
      base mpls-label-special-purpose-value;
      description
        "This identity represents the Extension Label.";
      reference
        "RFC7274: Allocating and Retiring Special-Purpose MPLS
         Labels.  Sections 3.1 and 5.";
    }

    /*** Collection of types related to routing ***/

    typedef router-id {
      type yang:dotted-quad;
      description
        "A 32-bit number in the dotted quad format assigned to each
         router. This number uniquely identifies the router within
         an Autonomous System.";
    }

    /*** Collection of types related to VPN ***/

    typedef route-target {
      type string {
        pattern
          '(0:(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
        +     '6[0-4][0-9]{3}|'
        +     '[0-5]?[0-9]{0,3}[0-9]):(429496729[0-5]|'
        +     '42949672[0-8][0-9]|'
        +     '4294967[01][0-9]{2}|429496[0-6][0-9]{3}|'
        +     '42949[0-5][0-9]{4}|'
```

```
          +     '4294[0-8][0-9]{5}|429[0-3][0-9]{6}|'
          +     '42[0-8][0-9]{7}|4[01][0-9]{8}|'
          +     '[0-3]?[0-9]{0,8}[0-9]))|'
          + '(1:((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|'
          +     '25[0-5])\.){3}([0-9]|[1-9][0-9]|'
          +     '1[0-9]{2}|2[0-4][0-9]|25[0-5])):(6553[0-5]|'
          +     '655[0-2][0-9]|'
          +     '65[0-4][0-9]{2}|6[0-4][0-9]{3}|'
          +     '[0-5]?[0-9]{0,3}[0-9]))|'
          + '(2:(429496729[0-5]|42949672[0-8][0-9]|'
          +     '4294967[01][0-9]{2}|'
          +     '429496[0-6][0-9]{3}|42949[0-5][0-9]{4}|'
          +     '4294[0-8][0-9]{5}|'
          +     '429[0-3][0-9]{6}|42[0-8][0-9]{7}|4[01][0-9]{8}|'
          +     '[0-3]?[0-9]{0,8}[0-9]):'
          +     '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
          +     '6[0-4][0-9]{3}|'
          +     '[0-5]?[0-9]{0,3}[0-9]))|'
          + '(6(:[a-fA-F0-9]{2}){6})|'
          + '(([3-57-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]{1,3}):'
          +     '[0-9a-fA-F]{1,12})';
        }
        description
          "A route target is an 8-octet BGP extended community
           initially identifying a set of sites in a BGP
           VPN (RFC 4364). However, it has since taken on a more
           general role in BGP route filtering.
           A route target consists of two or three fields:
           a 2-octet type field, an administrator field,
           and, optionally, an assigned number field.

           According to the data formats for type 0, 1, 2, and 6
           defined in RFC4360, RFC5668, and RFC7432, the encoding
           pattern is defined as:

           0:2-octet-asn:4-octet-number
           1:4-octet-ipv4addr:2-octet-number
           2:4-octet-asn:2-octet-number.
           6:6-octet-mac-address.

           Additionally, a generic pattern is defined for future
           route target types:

           2-octet-other-hex-number:6-octet-hex-number

           Some valid examples are: 0:100:100, 1:1.1.1.1:100,
           2:1234567890:203 and 6:26:00:08:92:78:00";
        reference
```

```
         "RFC4360: BGP Extended Communities Attribute.
          RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs)
          RFC5668: 4-Octet AS Specific BGP Extended Community.
          RFC7432: BGP MPLS-Based Ethernet VPN";
     }

     typedef ipv6-route-target {
       type string {
         pattern
             '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
             + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
             + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
             + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
             + ':'
             + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
             + '6[0-4][0-9]{3}|'
             + '[0-5]?[0-9]{0,3}[0-9])';
         pattern '((([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|'
             + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?))'
             + ':'
             + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
             + '6[0-4][0-9]{3}|'
             + '[0-5]?[0-9]{0,3}[0-9])';
       }
       description
         "An IPv6 route target is a 20-octet BGP IPv6 address
          specific extended community serving the same function
          as a standard 8-octet route target only allowing for
          an IPv6 address as the global administrator. The format
          is <ipv6-address:2-octet-number>.

          Some valid examples are: 2001:DB8::1:6544 and
          2001:DB8::5eb1:791:6b37:17958";
       reference
         "RFC5701: IPv6 Address Specific BGP Extended Community
                   Attribute";
     }

     typedef route-target-type {
       type enumeration {
         enum "import" {
           value 0;
           description
             "The route target applies to route import.";
         }
         enum "export" {
           value 1;
           description
```

```
             "The route target applies to route export.";
         }
         enum "both" {
           value 2;
           description
             "The route target applies to both route import and
              route export.";
         }
       }
       description
         "Indicates the role a route target takes
          in route filtering.";
       reference "RFC4364: BGP/MPLS IP Virtual Private Networks
                  (VPNs).";
     }

     typedef route-distinguisher {
       type string {
         pattern
           '(0:(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
         +     '6[0-4][0-9]{3}|'
         +     '[0-5]?[0-9]{0,3}[0-9]):(429496729[0-5]|'
         +     '42949672[0-8][0-9]|'
         +     '4294967[01][0-9]{2}|429496[0-6][0-9]{3}|'
         +     '42949[0-5][0-9]{4}|'
         +     '4294[0-8][0-9]{5}|429[0-3][0-9]{6}|'
         +     '42[0-8][0-9]{7}|4[01][0-9]{8}|'
         +     '[0-3]?[0-9]{0,8}[0-9]))|'
         + '(1:((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|'
         +     '25[0-5])\.){3}([0-9]|[1-9][0-9]|'
         +     '1[0-9]{2}|2[0-4][0-9]|25[0-5])):(6553[0-5]|'
         +     '655[0-2][0-9]|'
         +     '65[0-4][0-9]{2}|6[0-4][0-9]{3}|'
         +     '[0-5]?[0-9]{0,3}[0-9]))|'
         + '(2:(429496729[0-5]|42949672[0-8][0-9]|'
         +     '4294967[01][0-9]{2}|'
         +     '429496[0-6][0-9]{3}|42949[0-5][0-9]{4}|'
         +     '4294[0-8][0-9]{5}|'
         +     '429[0-3][0-9]{6}|42[0-8][0-9]{7}|4[01][0-9]{8}|'
         +     '[0-3]?[0-9]{0,8}[0-9]):'
         +     '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
         +     '6[0-4][0-9]{3}|'
         +     '[0-5]?[0-9]{0,3}[0-9]))|'
         + '(6(:[a-fA-F0-9]{2}){6})|'
         + '(([3-57-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]{1,3}):'
         +     '[0-9a-fA-F]{1,12})';
       }
       description
```

```
         "A route distinguisher is an 8-octet value used to
          distinguish routes from different BGP VPNs (RFC 4364).
          As per RFC 4360, a route distinguisher will have the same
          format as a route target and will consist of two or three
          fields including a 2-octet type field, an administrator
          field, and, optionally, an assigned number field.

          According to the data formats for type 0, 1, 2, and 6
          defined in RFC4360, RFC5668, and RFC7432, the encoding
          pattern is defined as:

          0:2-octet-asn:4-octet-number
          1:4-octet-ipv4addr:2-octet-number
          2:4-octet-asn:2-octet-number.
          6:6-octet-mac-address.

          Additionally, a generic pattern is defined for future
          route discriminator types:

          2-octet-other-hex-number:6-octet-hex-number

          Some valid examples are: 0:100:100, 1:1.1.1.1:100,
          2:1234567890:203 and 6:26:00:08:92:78:00";
       reference
         "RFC4360: BGP Extended Communities Attribute.
          RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs)
          RFC5668: 4-Octet AS Specific BGP Extended Community.
          RFC7432: BGP MPLS-Based Ethernet VPN";
     }

     typedef route-origin {
       type string {
         pattern
           '(0:(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
         +     '6[0-4][0-9]{3}|'
         +     '[0-5]?[0-9]{0,3}[0-9]):(429496729[0-5]|'
         +     '42949672[0-8][0-9]|'
         +     '4294967[01][0-9]{2}|429496[0-6][0-9]{3}|'
         +     '42949[0-5][0-9]{4}|'
         +     '4294[0-8][0-9]{5}|429[0-3][0-9]{6}|'
         +     '42[0-8][0-9]{7}|4[01][0-9]{8}|'
         +     '[0-3]?[0-9]{0,8}[0-9]))|'
         + '(1:((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|'
         +     '25[0-5])\.){3}([0-9]|[1-9][0-9]|'
         +     '1[0-9]{2}|2[0-4][0-9]|25[0-5])):(6553[0-5]|'
         +     '655[0-2][0-9]|'
         +     '65[0-4][0-9]{2}|6[0-4][0-9]{3}|'
         +     '[0-5]?[0-9]{0,3}[0-9]))|'
```

```
          + '(2:(429496729[0-5]|42949672[0-8][0-9]|'
          +     '4294967[01][0-9]{2}|'
          +     '429496[0-6][0-9]{3}|42949[0-5][0-9]{4}|'
          +     '4294[0-8][0-9]{5}|'
          +     '429[0-3][0-9]{6}|42[0-8][0-9]{7}|4[01][0-9]{8}|'
          +     '[0-3]?[0-9]{0,8}[0-9]):'
          +     '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
          +     '6[0-4][0-9]{3}|'
          +     '[0-5]?[0-9]{0,3}[0-9]))|'
          + '(6(:[a-fA-F0-9]{2}){6})|'
          + '(([3-57-9a-fA-F]|[1-9a-fA-F][0-9a-fA-F]{1,3}):'
          +    '[0-9a-fA-F]{1,12})';
        }
        description
          "A route origin is an 8-octet BGP extended community
           identifying the set of sites where the BGP route
           originated (RFC 4364). A route target consists of two
           or three fields: a 2-octet type field, an administrator
           field, and, optionally, an assigned number field.

           According to the data formats for type 0, 1, 2, and 6
           defined in RFC4360, RFC5668, and RFC7432, the encoding
           pattern is defined as:

           0:2-octet-asn:4-octet-number
           1:4-octet-ipv4addr:2-octet-number
           2:4-octet-asn:2-octet-number.
           6:6-octet-mac-address.

           Additionally, a generic pattern is defined for future
           route origin types:

           2-octet-other-hex-number:6-octet-hex-number

           Some valid examples are: 0:100:100, 1:1.1.1.1:100,
           2:1234567890:203 and 6:26:00:08:92:78:00";
        reference
          "RFC4360: BGP Extended Communities Attribute.
           RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs)
           RFC5668: 4-Octet AS Specific BGP Extended Community.
           RFC7432: BGP MPLS-Based Ethernet VPN";
      }

      typedef ipv6-route-origin {
        type string {
          pattern
              '((:|[0-9a-fA-F]{0,4}):)([0-9a-fA-F]{0,4}:){0,5}'
              + '((([0-9a-fA-F]{0,4}:)?(:|[0-9a-fA-F]{0,4}))|'
```

```
            + '(((25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])\.){3}'
            + '(25[0-5]|2[0-4][0-9]|[01]?[0-9]?[0-9])))'
            + ':'
            + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
            + '6[0-4][0-9]{3}|'
            + '[0-5]?[0-9]{0,3}[0-9])';
        pattern '((([^:]+:){6}(([^:]+:[^:]+)|(.*\..*)))|'
            + '((([^:]+:)*[^:]+)?::(([^:]+:)*[^:]+)?))'
            + ':'
            + '(6553[0-5]|655[0-2][0-9]|65[0-4][0-9]{2}|'
            + '6[0-4][0-9]{3}|'
            + '[0-5]?[0-9]{0,3}[0-9])';
      }
      description
        "An IPv6 route origin is a 20-octet BGP IPv6 address
         specific extended community serving the same function
         as a standard 8-octet route only allowing for
         an IPv6 address as the global administrator. The format
         is <ipv6-address:2-octet-number>.

         Some valid examples are: 2001:DB8::1:6544 and
         2001:DB8::5eb1:791:6b37:17958";
      reference
        "RFC5701: IPv6 Address Specific BGP Extended Community
                  Attribute";
    }

    /*** Collection of types common to multicast ***/

    typedef ipv4-multicast-group-address {
      type inet:ipv4-address {
        pattern '(2((2[4-9])|(3[0-9]))\.).*';
      }
      description
        "This type represents an IPv4 multicast group address,
         which is in the range from 224.0.0.0 to 239.255.255.255.";
      reference "RFC1112: Host Extensions for IP Multicasting.";
    }

    typedef ipv6-multicast-group-address {
      type inet:ipv6-address {
        pattern
          '(([fF]{2}[0-9a-fA-F]{2}):).*';
      }
      description
        "This type represents an IPv6 multicast group address,
         which is in the range of FF00::/8.";
      reference
```

```
         "RFC4291: IP Version 6 Addressing Architecture. Sec 2.7.
          RFC7346: IPv6 Multicast Address Scopes.";
     }

     typedef ip-multicast-group-address {
       type union {
         type ipv4-multicast-group-address;
         type ipv6-multicast-group-address;
       }
       description
         "This type represents a version-neutral IP multicast group
          address. The format of the textual representation implies
          the IP version.";
     }

     typedef ipv4-multicast-source-address {
       type union {
         type enumeration {
           enum "*" {
             description
               "Any source address.";
           }
         }
         type inet:ipv4-address;
       }
       description
         "Multicast source IPv4 address type.";
     }

     typedef ipv6-multicast-source-address {
       type union {
         type enumeration {
           enum "*" {
             description
               "Any source address.";
           }
         }
         type inet:ipv6-address;
       }
       description
         "Multicast source IPv6 address type.";
     }

     /*** Collection of types common to protocols ***/

     typedef bandwidth-ieee-float32 {
       type string {
         pattern
```

```
          '0[xX](0((\.0?)?[pP](\+)?0?|(\.0?))|'
        + '1(\.([0-9a-fA-F]{0,5}[02468aAcCeE]?)?)?[pP](\+)?(12[0-7]|'
        + '1[01][0-9]|0?[0-9]?[0-9])?)';
      }
      description
        "Bandwidth in IEEE 754 floating point 32-bit binary format:
         (-1)**(S) * 2**(Exponent-127) * (1 + Fraction),
         where Exponent uses 8 bits, and Fraction uses 23 bits.
         The units are octets per second.
         The encoding format is the external hexadecimal-significant
         character sequences specified in IEEE 754 and C99. The
         format is restricted to be normalized, non-negative, and
         non-fraction: 0x1.hhhhhhp{+}d or 0X1.HHHHHHP{+}D
         where 'h' and 'H' are hexadecimal digits, 'd' and 'D' are
         integers in the range of [0..127].
         When six hexadecimal digits are used for 'hhhhhh' or
         'HHHHHH', the least significant digit must be an even
         number. 'x' and 'X' indicate hexadecimal; 'p' and 'P'
         indicate power of two. Some examples are: 0x0p0, 0x1p10, and
         0x1.abcde2p+20";
      reference
        "IEEE Std 754-2008: IEEE Standard for Floating-Point
         Arithmetic.";
    }

    typedef link-access-type {
      type enumeration {
        enum "broadcast" {
          description
            "Specify broadcast multi-access network.";
        }
        enum "non-broadcast-multiaccess" {
          description
            "Specify Non-Broadcast Multi-Access (NBMA) network.";
        }
        enum "point-to-multipoint" {
          description
            "Specify point-to-multipoint network.";
        }
        enum "point-to-point" {
          description
            "Specify point-to-point network.";
        }
      }
      description
        "Link access type.";
    }
```

```
typedef timer-multiplier {
  type uint8;
  description
    "The number of timer value intervals that should be
     interpreted as a failure.";
}

typedef timer-value-seconds16 {
  type union {
    type uint16 {
      range "1..65535";
    }
    type enumeration {
      enum "infinity" {
        description
          "The timer is set to infinity.";
      }
      enum "not-set" {
        description
          "The timer is not set.";
      }
    }
  }
  units "seconds";
  description
    "Timer value type, in seconds (16-bit range).";
}

typedef timer-value-seconds32 {
  type union {
    type uint32 {
      range "1..4294967295";
    }
    type enumeration {
      enum "infinity" {
        description
          "The timer is set to infinity.";
      }
      enum "not-set" {
        description
          "The timer is not set.";
      }
    }
  }
  units "seconds";
  description
    "Timer value type, in seconds (32-bit range).";
}
```

```
typedef timer-value-milliseconds {
  type union {
    type uint32 {
      range "1..4294967295";
    }
    type enumeration {
      enum "infinity" {
        description
          "The timer is set to infinity.";
      }
      enum "not-set" {
        description
          "The timer is not set.";
      }
    }
  }
  units "milliseconds";
  description
    "Timer value type, in milliseconds.";
}

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

typedef timeticks64 {
  type uint64;
  description
    "This type is based on the timeticks type defined in
     RFC 6991, but with 64-bit width.  It represents the time,
     modulo 2^64, in hundredths of a second between two epochs.";
  reference "RFC 6991 - Common YANG Data Types";
}

typedef uint24 {
  type uint32 {
    range "0 .. 16777215";
  }
  description
    "24-bit unsigned integer";
}

/*** Collection of types related to MPLS/GMPLS ***/
```

```
typedef generalized-label {
  type binary;
  description
    "Generalized label. Nodes sending and receiving the
     Generalized Label are aware of the link-specific
     label context and type.";
  reference "RFC3471: Section 3.2";
}

typedef mpls-label-special-purpose {
  type identityref {
    base mpls-label-special-purpose-value;
  }
  description
    "This type represents the special-purpose Multiprotocol Label
     Switching (MPLS) label values.";
  reference
    "RFC3032: MPLS Label Stack Encoding.
     RFC7274: Allocating and Retiring Special-Purpose MPLS
     Labels.";
}

typedef mpls-label-general-use {
  type uint32 {
    range "16..1048575";
  }
  description
    "The 20-bit label values in an MPLS label stack entry,
     specified in RFC3032. This label value does not include
     the encodings of Traffic Class and TTL (time to live).
     The label range specified by this type is for general use,
     with special-purpose MPLS label values excluded.";
  reference "RFC3032: MPLS Label Stack Encoding.";
}

typedef mpls-label {
  type union {
    type mpls-label-special-purpose;
    type mpls-label-general-use;
  }
  description
    "The 20-bit label values in an MPLS label stack entry,
     specified in RFC3032. This label value does not include
     the encodings of Traffic Class and TTL (time to live).";
  reference "RFC3032: MPLS Label Stack Encoding.";
}

/*** Groupings **/
```

```
     grouping mpls-label-stack {
       description
         "This grouping specifies an MPLS label stack.  The label
          stack is encoded as a list of label stack entries.  The
          list key is an identifier which indicates relative
          ordering of each entry, with the lowest value identifier
          corresponding to the top of the label stack.";
       container mpls-label-stack {
         description
           "Container for a list of MPLS label stack entries.";
         list entry {
           key "id";
           description
             "List of MPLS label stack entries.";
           leaf id {
             type uint8;
             description
               "Identifies the entry in a sequence of MPLS label
                stack entries. An entry with a smaller identifier
                value precedes an entry with a larger identifier
                value in the label stack. The value of this ID has
                no semantic meaning other than relative ordering
                and referencing the entry.";
           }
           leaf label {
             type rt-types:mpls-label;
             description
               "Label value.";
           }
           leaf ttl {
             type uint8;
             description
               "Time to Live (TTL).";
             reference "RFC3032: MPLS Label Stack Encoding.";
           }
           leaf traffic-class {
             type uint8 {
               range "0..7";
             }
             description
               "Traffic Class (TC).";
             reference
               "RFC5462: Multiprotocol Label Switching (MPLS) Label
                Stack Entry: 'EXP' Field Renamed to 'Traffic Class'
                Field.";
           }
         }
       }
```

```
     }

     grouping vpn-route-targets {
       description
         "A grouping that specifies Route Target import-export rules
          used in the BGP enabled Virtual Private Networks (VPNs).";
       reference
         "RFC4364: BGP/MPLS IP Virtual Private Networks (VPNs).
          RFC4664: Framework for Layer 2 Virtual Private Networks
          (L2VPNs)";
       list vpn-target {
         key "route-target";
         description
           "List of Route Targets.";
         leaf route-target {
           type rt-types:route-target;
           description
             "Route Target value";
         }
         leaf route-target-type {
           type rt-types:route-target-type;
           mandatory true;
           description
             "Import/export type of the Route Target.";
         }
       }
     }
   }

   <CODE ENDS>
```

4. IANA Routing Types YANG Module

```
   <CODE BEGINS> file "iana-routing-types@2017-09-19.yang"
   module iana-routing-types {
     namespace "urn:ietf:params:xml:ns:yang:iana-routing-types";
     prefix iana-rt-types;

     organization
       "IANA";
     contact
       "        Internet Assigned Numbers Authority

        Postal: ICANN
                4676 Admiralty Way, Suite 330
                Marina del Rey, CA 90292

        Tel:    +1 310 823 9358
```

```
        <mailto:iana@iana.org>";
     description
       "This module contains a collection of YANG data types
        considered defined by IANA and used for routing
        protocols.

        Copyright (c) 2017 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
        (http://trustee.ietf.org/license-info).

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

     revision 2017-09-19 {
       description "Initial revision.";
       reference "RFC TBD: IANA Routing YANG Data Types";
     }

     /*** Collection of IANA types related to routing ***/
     /*** IANA address family enumeration ***/

     typedef address-family {
       type enumeration {
         enum ipv4 {
           value 1;
           description "IPv4 Address Family";
         }

         enum ipv6 {
           value 2;
           description "IPv6 Address Family";
         }

         enum nsap {
           value 3;
           description "OSI Network Service Access Point (NSAP)
                        Address Family";
         }

         enum hdlc {
           value 4;
```

```
            description "High-Level Data Link Control (HDLC)
                         Address Family";

          }

          enum bbn1822 {
            value 5;
            description "Bolt, Beranek, and Newman Report
                         1822 (BBN 1822) Address Family";
          }

          enum ieee802 {
            value 6;
            description "IEEE 802 Committee Address Family (aka,
                         MAC address)";
          }

          enum e163 {
            value 7;
            description "ITU-T E.163 Address Family";
          }

          enum e164 {
            value 8;
            description "ITU-T E.164 (SMDS, Frame Relay, ATM)
                         Address Family";
          }

          enum f69 {
            value 9;
            description "ITU-T F.69 (Telex) Address Family";
          }

          enum x121 {
            value 10;
            description  "ITU-T X.121 (X.25, Frame Relay)
                          Address Family";
          }

          enum ipx {
            value 11;
            description "Novell Internetwork Packet Exchange (IPX)
                         Address Family";
          }

          enum appletalk {
            value 12;
            description  "Apple AppleTalk Address Family";
```

```
         }

         enum decnet-iv {
           value 13;
           description "Digital Equipment DECnet Phase IV
                        Address Family";
         }

         enum vines {
           value 14;
           description "Banyan Vines Address Family";
         }

         enum e164-nsap {
           value 15;
           description "ITU-T E.164 with NSAP sub-address
                        Address Family";
         }

         enum dns {
           value 16;
           description "Domain Name System (DNS) Address
                        Family";
         }

         enum distinguished-name {
           value 17;
           description "Distinguished Name Address Family";
         }

         enum as-num {
           value 18;
           description "AS Number Address Family";
         }

         enum xtp-v4 {
           value 19;
           description "Xpress Transport Protocol (XTP) over IPv4
                        Address Family";
         }

         enum xtp-v6 {
           value 20;
           description "Xpress Transport Protocol (XTP) over IPv6
                        Address Family";
         }

         enum xtp-native {
```

```
            value 21;
            description "Xpress Transport Protocol (XTP) native mode
                         Address Family";
          }

          enum fc-port {
            value 22;
            description "Fibre Channel (FC) World-Wide Port Name
                          Address Family";
          }

          enum fc-node {
            value 23;
            description "Fibre Channel (FC) World-Wide Node Name
                         Address Family";
          }

          enum gwid {
            value 24;
            description
              "ATM Gateway Identifier (GWID) Number Address Family";
          }

          enum l2vpn {
            value 25;
            description "Layer-2 VPN (L2VPN) Address Family";
          }

          enum mpls-tp-section-eid {
            value 26;
            description "MPLS-TP Section Endpoint Identifier
                         Address Family";
          }

          enum mpls-tp-lsp-eid {
            value 27;
            description "MPLS-TP LSP Endpoint Identifier
                         Address Family";
          }

          enum mpls-tp-pwe-eid {
            value 28;
            description "MPLS-TP Pseudowire Endpoint Identifier
                         Address Family";
          }

          enum mt-v4 {
            value 29;
```

```
          description "Multi-Topology IPv4 Address Family";
        }

        enum mt-v6 {
          value 30;
          description "Multi-Topology IPv6 Address Family";
        }

        enum eigrp-common-sf {
          value 16384;
          description "Enhanced Interior Gateway Routing Protocol
                       (EIGRP) Common Service Family Address
                       Family";
        }

        enum eigrp-v4-sf {
          value 16385;
          description "Enhanced Interior Gateway Routing Protocol
                       (EIGRP) IPv4 Service Family Address Family";
        }

        enum eigrp-v6-sf {
          value 16386;
          description "Enhanced Interior Gateway Routing Protocol
                      (EIGRP) IPv6 Service Family Address Family";
        }

        enum lcaf {
          value 16387;
          description "LISP Canonical Address Format (LCAF)
                       Address Family";
        }

        enum bgp-ls {
          value 16388;
          description "Border Gateway Protocol - Link State (BGP-LS)
                       Address Family";
        }

        enum mac-48 {
          value 16389;
          description  "IEEE 48-bit Media Access Control (MAC)
                        Address Family";
        }

        enum mac-64 {
          value 16390;
          description "IEEE 64-bit Media Access Control (MAC)
```

```
                        Address Family";
        }

        enum trill-oui {
          value 16391;
          description "TRILL IEEE Organizationally Unique
                       Identifier (OUI) Address Family";
        }

        enum trill-mac-24 {
          value 16392;
          description "TRILL Final 3 octets of 48-bit MAC
                       address Address Family";
        }

        enum trill-mac-40 {
          value 16393;
          description "TRILL Final 5 octets of 64-bit MAC
                       address Address Family";
        }

        enum ipv6-64 {
          value 16394;
          description "First 8 octets (64-bits) of an IPv6
                       address Address Family";
        }

        enum trill-rbridge-port-id {
          value 16395;
          description "TRILL Remote Bridge (RBridge) Port ID
                       Address Family";
        }

        enum trill-nickname {
          value 16396;
          description "TRILL Nickname Address Family";
        }
      }
      description "Enumeration containing all the IANA
                   defined address families.";

    }

    /*** SAFIs for Multi-Protocol BGP enumeration ***/

    typedef bgp-safi {
      type enumeration {
        enum unicast-safi {
```

```
          value 1;
          description "Unicast SAFI";
        }

        enum multicast-safi {
          value 2;
          description "Multicast SAFI";
        }

        enum labeled-unicast-safi {
          value 4;
          description "Labeled Unicast SAFI";
        }

        enum multicast-vpn-safi {
          value 5;
          description "Multicast VPN SAFI";
        }

        enum pseudowire-safi {
          value 6;
          description "Multi-segment Pseudowire VPN SAFI";
        }

        enum tunnel-encap-safi {
          value 7;
          description "Tunnel Encap SAFI";
        }

        enum mcast-vpls-safi {
          value 8;
          description "Multicast Virtual Private LAN Service
                       (VPLS) SAFI";
        }

        enum tunnel-safi {
          value 64;
          description "Tunnel SAFI";
        }

        enum vpls-safi {
          value 65;
          description "Virtual Private LAN Service (VPLS) SAFI";
        }

        enum mdt-safi {
          value 66;
          description "Multicast Distribution Tree (MDT) SAFI";
```

```
        }

        enum v4-over-v6-safi {
          value 67;
          description "IPv4 over IPv6 SAFI";
        }

        enum v6-over-v4-safi {
          value 68;
          description "IPv6 over IPv4 SAFI";
        }

        enum l1-vpn-auto-discovery-safi {
          value 69;
          description "Layer-1 VPN Auto Discovery SAFI";
        }

        enum evpn-safi {
          value 70;
          description "Ethernet VPN (EVPN) SAFI";
        }

        enum bgp-ls-safi {
          value 71;
          description "BGP Link-State (BGP-LS) SAFI";
        }

        enum bgp-ls-vpn-safi {
          value 72;
          description "BGP Link-State (BGP-LS) VPN SAFI";
        }

        enum sr-te-safi {
          value 73;
          description "Segment Routing - Traffic Engineering
                       (SR-TE) SAFI";
        }

        enum labeled-vpn-safi {
          value 128;
          description "MPLS Labeled VPN SAFI";
        }

        enum multicast-mpls-vpn-safi {
          value 129;
          description "Multicast for BGP/MPLS IP VPN SAFI";
        }
```

```
        enum route-target-safi {
          value 132;
          description "Route Target SAFI";
        }

        enum ipv4-flow-spec-safi {
          value 133;
          description "IPv4 Flow Specification SAFI";
        }

        enum vpnv4-flow-spec-safi {
          value 134;
          description "IPv4 VPN Flow Specification SAFI";
        }

        enum vpn-auto-discovery-safi {
          value 140;
          description "VPN Auto-Discovery SAFI";
        }
      }
      description "Enumeration for BGP Subsequent Address
                   Family Identifier (SAFI) - RFC 4760.";
    }
  }

<CODE ENDS>
```

5. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

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

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

This document registers the following YANG modules in the YANG Module Names registry [RFC6020]:

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

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

5.1. IANA-Maintained iana-routing-types Module

This document defines the initial version of the IANA-maintained iana-routing-types YANG module.

The iana-routing-types YANG module is intended to reflect the "Address Family Numbers" registry [IANA-ADDRESS-FAMILY-REGISTRY] and "Subsequent Address Family Identifiers (SAFI) Parameters" registry [IANA-SAFI-REGISTRY].

IANA has added this notes to the "iana-routing-types YANG Module" registry:

> Address Families and Subsequent Address Families must not be directly added to the iana-routing-types YANG module. They must instead be respectively added to the "Address Family Numbers" and "Subsequent Address Family Identifiers (SAFI) Parameters" registries.

When an Address Family or Subsequent Address Family is respectively added to the "Address Family Numbers" registry or the "Subsequent Address Family Identifiers (SAFI) Parameters" registry, a new "enum" statement must be added to the iana-routing-types YANG module. The name of the "enum" is the same as the corresponding address family or SAFI only it will be a valid YANG identifier in all lowercase and with hyphens separating individual words in compound identifiers. The following substatements to the "enum" statement should be defined:

"val": Contains the IANA assigned value corresponding to the address-family or "bgp-safi" for subsequent address families.

"status": Include only if a registration has been deprecated (use the value "deprecated") or obsoleted (use the value "obsolete").

"description": Replicate the description from the registry, if any. Insert line breaks as needed so that the line does not exceed 72 characters.

"reference": Replicate the reference from the registry, if any, and add the title of the document.

Unassigned or reserved values are not present in these modules.

When the iana-routing-types YANG module is updated, a new "revision" statement must be added in front of the existing revision statements.

IANA has added this new note to the "Address Family Numbers" and "Subsequent Address Family Identifiers (SAFI) Parameters" registries:

> When this registry is modified, the YANG module iana-routing-types must be updated as defined in RFC XXXX.

6. Security Considerations

This document defines common data types using the YANG data modeling language. The definitions themselves have no security impact on the Internet, but the usage of these definitions in concrete YANG modules might have. The security considerations spelled out in the YANG specification [RFC7950] apply for this document as well.

7. Acknowledgements

The Routing Area Yang Architecture design team members included Acee Lindem, Anees Shaikh, Christian Hopps, Dean Bogdanovic, Ebben Aries, Lou Berger, Qin Wu, Rob Shakir, Xufeng Liu, and Yingzhen Qu.

Thanks to Martin Bjorkland, Tom Petch, Stewart Bryant, and Radek Krejci for comments on the model and document text. Thanks to Jeff Haas and Robert Raszuk for suggestions for additional common routing types.

8. References

8.1. Normative References

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

[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>.

[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>.

[IANA-ADDRESS-FAMILY-REGISTRY] "IANA Address Family Registry", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml#address-family-numbers-2>.

[IANA-SAFI-REGISTRY] "IANA Subsequent Address Family Identities (SAFI) Parameters Registry", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml#safi-namespace-2>.

8.2. Informative References

[IEEE754] IEEE, "IEEE Standard for Floating-Point Arithmetic", IEEE Std 754-2008, August 2008.

[I-D.ietf-bfd-yang] Rahman, R., Zheng, L., Jethanandani, M., Networks, J., and G. Mirsky, "YANG Data Model for Bidirectional Forwarding Detection (BFD)", draft-ietf-bfd-yang-06 (work in progress), June 2017.

[I-D.ietf-idr-bgp-model] Shaikh, A., Shakir, R., Patel, K., Hares, S., D'Souza, K., Bansal, D., Clemm, A., Zhdankin, A., Jethanandani, M., and X. Liu, "BGP Model for Service Provider Networks", draft-ietf-idr-bgp-model-02 (work in progress), July 2016.

[I-D.ietf-ospf-yang]
Yeung, D., Qu, Y., Zhang, Z., Chen, I., and A. Lindem, "Yang Data Model for OSPF Protocol", draft-ietf-ospf-yang-08 (work in progress), July 2017.

[I-D.ietf-pim-yang]
Liu, X., McAllister, P., Peter, A., Sivakumar, M., Liu, Y., and f. hu, "A YANG data model for Protocol-Independent Multicast (PIM)", draft-ietf-pim-yang-08 (work in progress), April 2017.

[I-D.ietf-teas-yang-rsvp]
Beeram, V., Saad, T., Gandhi, R., Liu, X., Bryskin, I., and H. Shah, "A YANG Data Model for Resource Reservation Protocol (RSVP)", draft-ietf-teas-yang-rsvp-07 (work in progress), March 2017.

[I-D.ietf-teas-yang-te]
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-08 (work in progress), July 2017.

[I-D.ietf-bess-l2vpn-yang]
Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-06 (work in progress), June 2017.

[I-D.ietf-bess-l3vpn-yang]
Jain, D., Patel, K., Brissette, P., Li, Z., Zhuang, S., Liu, X., Haas, J., Esale, S., and B. Wen, "Yang Data Model for BGP/MPLS L3 VPNs", draft-ietf-bess-l3vpn-yang-01 (work in progress), April 2017.

[I-D.ietf-mpls-base-yang]
Raza, K., Gandhi, R., Liu, X., Beeram, V., Saad, T., Bryskin, I., Chen, X., Jones, R., and B. Wen, "A YANG Data Model for MPLS Base", draft-ietf-mpls-base-yang-05 (work in progress), July 2017.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

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

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <https://www.rfc-editor.org/info/rfc4664>.

[RFC5701] Rekhter, Y., "IPv6 Address Specific BGP Extended Community Attribute", RFC 5701, DOI 10.17487/RFC5701, November 2009, <https://www.rfc-editor.org/info/rfc5701>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC7274] Kompella, K., Andersson, L., and A. Farrel, "Allocating and Retiring Special-Purpose MPLS Labels", RFC 7274, DOI 10.17487/RFC7274, June 2014, <https://www.rfc-editor.org/info/rfc7274>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <https://www.rfc-editor.org/info/rfc7432>.

Authors' Addresses

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: Xufeng_Liu@jabil.com

Yingzhen Qu
Futurewei Technologies, Inc.
2330 Central Expressway
Santa Clara CA 95050
USA

EMail: yingzhen.qu@huawei.com

Acee Lindem
Cisco Systems
301 Midenhall Way
Cary, NC 27513
USA

EMail: acee@cisco.com

Christian Hopps
Deutsche Telekom

EMail: chopps@chopps.org

Lou Berger
LabN Consulting, L.L.C.

EMail: lberger@labn.net