Internet Engineering Task Force (IETF)
Request for Comments: 8344
Obsoletes: 7277
Category: Standards Track
ISSN: 2070-1721

M. Bjorklund
Tail-f Systems

March 2018

# A YANG Data Model for IP Management

## Abstract

This document defines a YANG data model for management of IP implementations. The data model includes configuration and system state.

The YANG data model in this document conforms to the Network Management Datastore Architecture defined in RFC 8342.

This document obsoletes RFC 7277.

## Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8344.

Table of Contents



## 1. Introduction

This document defines a YANG data model [RFC7950] for management of IP implementations.

The data model covers configuration of per-interface IPv4 and IPv6 parameters as well as mappings of IP addresses to link-layer addresses. It also provides information about which IP addresses are operationally used and which link-layer mappings exist. Per-interface parameters are added through augmentation of the interface data model defined in [RFC8343].

This version of the IP data model supports the Network Management Datastore Architecture (NMDA) [RFC8342].

### 1.1. Summary of Changes from RFC 7277

The "ipv4" and "ipv6" subtrees with "config false" data nodes in the "/interfaces-state/interface" subtree are deprecated. All "config false" data nodes are now present in the "ipv4" and "ipv6" subtrees in the "/interfaces/interface" subtree.

Servers that do not implement NMDA or that wish to support clients that do not implement NMDA MAY implement the deprecated "ipv4" and "ipv6" subtrees in the "/interfaces-state/interface" subtree.

## 1.2. Terminology

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

The following terms are defined in [RFC8342] and are not redefined here:

- client
- server
- configuration
- system state
- intended configuration
- running configuration datastore
- operational state
- operational state datastore

The following terms are defined in [RFC7950] and are not redefined here:

- augment
- data model
- data node

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

## 1.3. Tree Diagrams

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

## 2. IP Data Model

This document defines the YANG module "ietf-ip", which augments the "interface" lists defined in the "ietf-interfaces" module [RFC8343] with IP-specific data nodes.

The data model has the following structure for IP data nodes per interface, excluding the deprecated data nodes:

```
module: ietf-ip
  augment /if:interfaces/if:interface:
    +--rw ipv4!
    |  +--rw enabled?      boolean
    |  +--rw forwarding?   boolean
    |  +--rw mtu?          uint16
    |  +--rw address* [ip]
    |  |  +--rw ip               inet:ipv4-address-no-zone
    |  |  +--rw (subnet)
    |  |  |  +--:(prefix-length)
    |  |  |  |  +--rw prefix-length?   uint8
    |  |  |  +--:(netmask)
    |  |  |     +--rw netmask?         yang:dotted-quad
    |  |  |              {ipv4-non-contiguous-netmasks}?
    |  |  +--ro origin?          ip-address-origin
    |  +--rw neighbor* [ip]
    |     +--rw ip                    inet:ipv4-address-no-zone
    |     +--rw link-layer-address    yang:phys-address
    |     +--ro origin?               neighbor-origin
    +--rw ipv6!
       +--rw enabled?                    boolean
       +--rw forwarding?                 boolean
       +--rw mtu?                        uint32
       +--rw address* [ip]
       |  +--rw ip               inet:ipv6-address-no-zone
       |  +--rw prefix-length    uint8
       |  +--ro origin?          ip-address-origin
       |  +--ro status?          enumeration
       +--rw neighbor* [ip]
       |  +--rw ip                    inet:ipv6-address-no-zone
       |  +--rw link-layer-address    yang:phys-address
       |  +--ro origin?               neighbor-origin
       |  +--ro is-router?            empty
       |  +--ro state?                enumeration
       +--rw dup-addr-detect-transmits?   uint32
```

```
      +--rw autoconf
         +--rw create-global-addresses?        boolean
         +--rw create-temporary-addresses?     boolean
         |       {ipv6-privacy-autoconf}?
         +--rw temporary-valid-lifetime?       uint32
         |       {ipv6-privacy-autoconf}?
         +--rw temporary-preferred-lifetime?   uint32
                 {ipv6-privacy-autoconf}?
```

The data model defines two containers per interface -- "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "enabled" that controls whether or not the address family is enabled on that interface, and a leaf "forwarding" that controls whether or not IP packet forwarding for the address family is enabled on the interface. In each container, there is also a list of addresses and a list of mappings from IP addresses to link-layer addresses.

## 3. Relationship to the IP-MIB

If the device implements the IP-MIB [RFC4293], each entry in the "ipv4/address" and "ipv6/address" lists is mapped to one ipAddressEntry, where the ipAddressIfIndex refers to the "address" entry's interface.

The IP-MIB defines objects to control IPv6 Router Advertisement messages. The corresponding YANG data nodes are defined in [RFC8022].

The entries in "ipv4/neighbor" and "ipv6/neighbor" are mapped to ipNetToPhysicalTable.

The following table lists the YANG data nodes with corresponding objects in the IP-MIB.

| YANG data node in /if:interfaces/if:interface | IP-MIB object |
|---|---|
| ipv4 | ipv4InterfaceEnableStatus |
| ipv4/enabled | ipv4InterfaceEnableStatus |
| ipv4/address | ipAddressEntry |
| ipv4/address/ip | ipAddressAddrType |
| | ipAddressAddr |
| ipv4/neighbor | ipNetToPhysicalEntry |
| ipv4/neighbor/ip | ipNetToPhysicalNetAddressType |
| | ipNetToPhysicalNetAddress |
| ipv4/neighbor/link-layer-address | ipNetToPhysicalPhysAddress |
| ipv4/neighbor/origin | ipNetToPhysicalType |
| ipv6 | ipv6InterfaceEnableStatus |
| ipv6/enabled | ipv6InterfaceEnableStatus |
| ipv6/forwarding | ipv6InterfaceForwarding |
| ipv6/address | ipAddressEntry |
| ipv6/address/ip | ipAddressAddrType |
| | ipAddressAddr |
| ipv4/address/origin | ipAddressOrigin |
| ipv6/address/status | ipAddressStatus |
| ipv6/neighbor | ipNetToPhysicalEntry |
| ipv6/neighbor/ip | ipNetToPhysicalNetAddressType |
| | ipNetToPhysicalNetAddress |
| ipv6/neighbor/link-layer-address | ipNetToPhysicalPhysAddress |
| ipv6/neighbor/origin | ipNetToPhysicalType |
| ipv6/neighbor/state | ipNetToPhysicalState |

YANG Interface Data Nodes and Related IP-MIB Objects

## 4. IP Management YANG Module

This module imports typedefs from [RFC6991] and [RFC8343], and it references [RFC791], [RFC826], [RFC4861], [RFC4862], [RFC4941], [RFC7217], and [RFC8200].

```
<CODE BEGINS> file "ietf-ip@2018-02-22.yang"
module ietf-ip {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-ip";
  prefix ip;

  import ietf-interfaces {
    prefix if;
  }
  import ietf-inet-types {
    prefix inet;
  }
  import ietf-yang-types {
    prefix yang;
  }

  organization
    "IETF NETMOD (Network Modeling) Working Group";

  contact
    "WG Web:   <https://datatracker.ietf.org/wg/netmod/>
     WG List:  <mailto:netmod@ietf.org>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>";
  description
    "This module contains a collection of YANG definitions for
     managing IP implementations.

     Copyright (c) 2018 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 8344; see
     the RFC itself for full legal notices.";
```

```
  revision 2018-02-22 {
    description
      "Updated to support NMDA.";
    reference
      "RFC 8344: A YANG Data Model for IP Management";
  }

  revision 2014-06-16 {
    description
      "Initial revision.";
    reference
      "RFC 7277: A YANG Data Model for IP Management";
  }

  /*
   * Features
   */

  feature ipv4-non-contiguous-netmasks {
    description
      "Indicates support for configuring non-contiguous
       subnet masks.";
  }

  feature ipv6-privacy-autoconf {
    description
      "Indicates support for privacy extensions for stateless address
       autoconfiguration in IPv6.";
    reference
      "RFC 4941: Privacy Extensions for Stateless Address
                 Autoconfiguration in IPv6";
  }

  /*
   * Typedefs
   */

  typedef ip-address-origin {
    type enumeration {
      enum other {
        description
          "None of the following.";
      }
```

```
        enum static {
          description
            "Indicates that the address has been statically
             configured -- for example, using the Network Configuration
             Protocol (NETCONF) or a command line interface.";
        }
        enum dhcp {
          description
            "Indicates an address that has been assigned to this
             system by a DHCP server.";
        }
        enum link-layer {
          description
            "Indicates an address created by IPv6 stateless
             autoconfiguration that embeds a link-layer address in its
             interface identifier.";
        }
        enum random {
          description
            "Indicates an address chosen by the system at
             random, e.g., an IPv4 address within 169.254/16, a
             temporary address as described in RFC 4941, or a
             semantically opaque address as described in RFC 7217.";
          reference
            "RFC 4941: Privacy Extensions for Stateless Address
                       Autoconfiguration in IPv6
             RFC 7217: A Method for Generating Semantically Opaque
                       Interface Identifiers with IPv6 Stateless
                       Address Autoconfiguration (SLAAC)";
        }
      }
      description
        "The origin of an address.";
    }

    typedef neighbor-origin {
      type enumeration {
        enum other {
          description
            "None of the following.";
        }
        enum static {
          description
            "Indicates that the mapping has been statically
             configured -- for example, using NETCONF or a command line
             interface.";
        }
```

```
        enum dynamic {
          description
            "Indicates that the mapping has been dynamically resolved
             using, for example, IPv4 ARP or the IPv6 Neighbor
             Discovery protocol.";
        }
      }
      description
        "The origin of a neighbor entry.";
    }

    /*
     * Data nodes
     */

    augment "/if:interfaces/if:interface" {
      description
        "IP parameters on interfaces.

         If an interface is not capable of running IP, the server
         must not allow the client to configure these parameters.";

      container ipv4 {
        presence
          "Enables IPv4 unless the 'enabled' leaf
           (which defaults to 'true') is set to 'false'";
        description
          "Parameters for the IPv4 address family.";

        leaf enabled {
          type boolean;
          default true;
          description
            "Controls whether IPv4 is enabled or disabled on this
             interface.  When IPv4 is enabled, this interface is
             connected to an IPv4 stack, and the interface can send
             and receive IPv4 packets.";
        }
        leaf forwarding {
          type boolean;
          default false;
          description
            "Controls IPv4 packet forwarding of datagrams received by,
             but not addressed to, this interface.  IPv4 routers
             forward datagrams.  IPv4 hosts do not (except those
             source-routed via the host).";
        }
```

```
        leaf mtu {
          type uint16 {
            range "68..max";
          }
          units "octets";
          description
            "The size, in octets, of the largest IPv4 packet that the
             interface will send and receive.

             The server may restrict the allowed values for this leaf,
             depending on the interface's type.

             If this leaf is not configured, the operationally used MTU
             depends on the interface's type.";
          reference
            "RFC 791: Internet Protocol";
        }
        list address {
          key "ip";
          description
            "The list of IPv4 addresses on the interface.";

          leaf ip {
            type inet:ipv4-address-no-zone;
            description
              "The IPv4 address on the interface.";
          }
          choice subnet {
            mandatory true;
            description
              "The subnet can be specified as a prefix length or,
               if the server supports non-contiguous netmasks, as
               a netmask.";
            leaf prefix-length {
              type uint8 {
                range "0..32";
              }
              description
                "The length of the subnet prefix.";
            }
            leaf netmask {
              if-feature ipv4-non-contiguous-netmasks;
              type yang:dotted-quad;
              description
                "The subnet specified as a netmask.";
            }
          }
```

```
          leaf origin {
            type ip-address-origin;
            config false;
            description
              "The origin of this address.";
          }
        }
        list neighbor {
          key "ip";
          description
            "A list of mappings from IPv4 addresses to
             link-layer addresses.

             Entries in this list in the intended configuration are
             used as static entries in the ARP Cache.

             In the operational state, this list represents the ARP
             Cache.";
          reference
            "RFC 826: An Ethernet Address Resolution Protocol";

          leaf ip {
            type inet:ipv4-address-no-zone;
            description
              "The IPv4 address of the neighbor node.";
          }
          leaf link-layer-address {
            type yang:phys-address;
            mandatory true;
            description
              "The link-layer address of the neighbor node.";
          }
          leaf origin {
            type neighbor-origin;
            config false;
            description
              "The origin of this neighbor entry.";
          }
        }
      }
```

```
      container ipv6 {
        presence
          "Enables IPv6 unless the 'enabled' leaf
           (which defaults to 'true') is set to 'false'";
        description
          "Parameters for the IPv6 address family.";

        leaf enabled {
          type boolean;
          default true;
          description
            "Controls whether IPv6 is enabled or disabled on this
             interface.  When IPv6 is enabled, this interface is
             connected to an IPv6 stack, and the interface can send
             and receive IPv6 packets.";
        }
        leaf forwarding {
          type boolean;
          default false;
          description
            "Controls IPv6 packet forwarding of datagrams received by,
             but not addressed to, this interface.  IPv6 routers
             forward datagrams.  IPv6 hosts do not (except those
             source-routed via the host).";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
                       Section 6.2.1, IsRouter";
        }
        leaf mtu {
          type uint32 {
            range "1280..max";
          }
          units "octets";
          description
            "The size, in octets, of the largest IPv6 packet that the
             interface will send and receive.

             The server may restrict the allowed values for this leaf,
             depending on the interface's type.

             If this leaf is not configured, the operationally used MTU
             depends on the interface's type.";
          reference
            "RFC 8200: Internet Protocol, Version 6 (IPv6)
                       Specification
                       Section 5";
        }
```

```
      list address {
        key "ip";
        description
          "The list of IPv6 addresses on the interface.";

        leaf ip {
          type inet:ipv6-address-no-zone;
          description
            "The IPv6 address on the interface.";
        }
        leaf prefix-length {
          type uint8 {
            range "0..128";
          }
          mandatory true;
          description
            "The length of the subnet prefix.";
        }
        leaf origin {
          type ip-address-origin;
          config false;
          description
            "The origin of this address.";
        }
        leaf status {
          type enumeration {
            enum preferred {
              description
                "This is a valid address that can appear as the
                 destination or source address of a packet.";
            }
            enum deprecated {
              description
                "This is a valid but deprecated address that should
                 no longer be used as a source address in new
                 communications, but packets addressed to such an
                 address are processed as expected.";
            }
            enum invalid {
              description
                "This isn't a valid address, and it shouldn't appear
                 as the destination or source address of a packet.";
            }
```

```
            enum inaccessible {
              description
                "The address is not accessible because the interface
                 to which this address is assigned is not
                 operational.";
            }
            enum unknown {
              description
                "The status cannot be determined for some reason.";
            }
            enum tentative {
              description
                "The uniqueness of the address on the link is being
                 verified.  Addresses in this state should not be
                 used for general communication and should only be
                 used to determine the uniqueness of the address.";
            }
            enum duplicate {
              description
                "The address has been determined to be non-unique on
                 the link and so must not be used.";
            }
            enum optimistic {
              description
                "The address is available for use, subject to
                 restrictions, while its uniqueness on a link is
                 being verified.";
            }
          }
          config false;
          description
            "The status of an address.  Most of the states correspond
             to states from the IPv6 Stateless Address
             Autoconfiguration protocol.";
          reference
            "RFC 4293: Management Information Base for the
                       Internet Protocol (IP)
                       - IpAddressStatusTC
             RFC 4862: IPv6 Stateless Address Autoconfiguration";
        }
      }
```

```
      list neighbor {
        key "ip";
        description
          "A list of mappings from IPv6 addresses to
           link-layer addresses.

           Entries in this list in the intended configuration are
           used as static entries in the Neighbor Cache.

           In the operational state, this list represents the
           Neighbor Cache.";
        reference
          "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)";

        leaf ip {
          type inet:ipv6-address-no-zone;
          description
            "The IPv6 address of the neighbor node.";
        }
        leaf link-layer-address {
          type yang:phys-address;
          mandatory true;
          description
            "The link-layer address of the neighbor node.

             In the operational state, if the neighbor's 'state' leaf
             is 'incomplete', this leaf is not instantiated.";
        }
        leaf origin {
          type neighbor-origin;
          config false;
          description
            "The origin of this neighbor entry.";
        }
        leaf is-router {
          type empty;
          config false;
          description
            "Indicates that the neighbor node acts as a router.";
        }
```

```
          leaf state {
            type enumeration {
              enum incomplete {
                description
                  "Address resolution is in progress, and the
                   link-layer address of the neighbor has not yet been
                   determined.";
              }
              enum reachable {
                description
                  "Roughly speaking, the neighbor is known to have been
                   reachable recently (within tens of seconds ago).";
              }
              enum stale {
                description
                  "The neighbor is no longer known to be reachable, but
                   until traffic is sent to the neighbor no attempt
                   should be made to verify its reachability.";
              }
              enum delay {
                description
                  "The neighbor is no longer known to be reachable, and
                   traffic has recently been sent to the neighbor.
                   Rather than probe the neighbor immediately, however,
                   delay sending probes for a short while in order to
                   give upper-layer protocols a chance to provide
                   reachability confirmation.";
              }
              enum probe {
                description
                  "The neighbor is no longer known to be reachable, and
                   unicast Neighbor Solicitation probes are being sent
                   to verify reachability.";
              }
            }
            config false;
            description
              "The Neighbor Unreachability Detection state of this
               entry.";
            reference
              "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
                         Section 7.3.2";
          }
        }
```

```
      leaf dup-addr-detect-transmits {
        type uint32;
        default 1;
        description
          "The number of consecutive Neighbor Solicitation messages
           sent while performing Duplicate Address Detection on a
           tentative address.  A value of zero indicates that
           Duplicate Address Detection is not performed on
           tentative addresses.  A value of one indicates a single
           transmission with no follow-up retransmissions.";
        reference
          "RFC 4862: IPv6 Stateless Address Autoconfiguration";
      }
      container autoconf {
        description
          "Parameters to control the autoconfiguration of IPv6
           addresses, as described in RFC 4862.";
        reference
          "RFC 4862: IPv6 Stateless Address Autoconfiguration";

        leaf create-global-addresses {
          type boolean;
          default true;
          description
            "If enabled, the host creates global addresses as
             described in RFC 4862.";
          reference
            "RFC 4862: IPv6 Stateless Address Autoconfiguration
                       Section 5.5";
        }
        leaf create-temporary-addresses {
          if-feature ipv6-privacy-autoconf;
          type boolean;
          default false;
          description
            "If enabled, the host creates temporary addresses as
             described in RFC 4941.";
          reference
            "RFC 4941: Privacy Extensions for Stateless Address
                       Autoconfiguration in IPv6";
        }
```

```
          leaf temporary-valid-lifetime {
            if-feature ipv6-privacy-autoconf;
            type uint32;
            units "seconds";
            default 604800;
            description
              "The time period during which the temporary address
               is valid.";
            reference
              "RFC 4941: Privacy Extensions for Stateless Address
                         Autoconfiguration in IPv6
                         - TEMP_VALID_LIFETIME";
          }
          leaf temporary-preferred-lifetime {
            if-feature ipv6-privacy-autoconf;
            type uint32;
            units "seconds";
            default 86400;
            description
              "The time period during which the temporary address is
               preferred.";
            reference
              "RFC 4941: Privacy Extensions for Stateless Address
                         Autoconfiguration in IPv6
                         - TEMP_PREFERRED_LIFETIME";
          }
        }
      }
    }
```

```
     /*
      * Legacy operational state data nodes
      */

     augment "/if:interfaces-state/if:interface" {
       status deprecated;
       description
         "Data nodes for the operational state of IP on interfaces.";

       container ipv4 {
         presence
           "Present if IPv4 is enabled on this interface";
         config false;
         status deprecated;
         description
           "Interface-specific parameters for the IPv4 address family.";

         leaf forwarding {
           type boolean;
           status deprecated;
           description
             "Indicates whether IPv4 packet forwarding is enabled or
              disabled on this interface.";
         }
         leaf mtu {
           type uint16 {
             range "68..max";
           }
           units "octets";
           status deprecated;
           description
             "The size, in octets, of the largest IPv4 packet that the
              interface will send and receive.";
           reference
             "RFC 791: Internet Protocol";
         }
         list address {
           key "ip";
           status deprecated;
           description
             "The list of IPv4 addresses on the interface.";

           leaf ip {
             type inet:ipv4-address-no-zone;
             status deprecated;
             description
               "The IPv4 address on the interface.";
           }
```

```
        choice subnet {
          status deprecated;
          description
            "The subnet can be specified as a prefix length or,
             if the server supports non-contiguous netmasks, as
             a netmask.";
          leaf prefix-length {
            type uint8 {
              range "0..32";
            }
            status deprecated;
            description
              "The length of the subnet prefix.";
          }
          leaf netmask {
            if-feature ipv4-non-contiguous-netmasks;
            type yang:dotted-quad;
            status deprecated;
            description
              "The subnet specified as a netmask.";
          }
        }
        leaf origin {
          type ip-address-origin;
          status deprecated;
          description
            "The origin of this address.";
        }
      }
      list neighbor {
        key "ip";
        status deprecated;
        description
          "A list of mappings from IPv4 addresses to
           link-layer addresses.

           This list represents the ARP Cache.";
        reference
          "RFC 826: An Ethernet Address Resolution Protocol";

        leaf ip {
          type inet:ipv4-address-no-zone;
          status deprecated;
          description
            "The IPv4 address of the neighbor node.";
        }
```

```
          leaf link-layer-address {
            type yang:phys-address;
            status deprecated;
            description
              "The link-layer address of the neighbor node.";
          }
          leaf origin {
            type neighbor-origin;
            status deprecated;
            description
              "The origin of this neighbor entry.";
          }
        }
      }

      container ipv6 {
        presence
          "Present if IPv6 is enabled on this interface";
        config false;
        status deprecated;
        description
          "Parameters for the IPv6 address family.";

        leaf forwarding {
          type boolean;
          default false;
          status deprecated;
          description
            "Indicates whether IPv6 packet forwarding is enabled or
             disabled on this interface.";
          reference
            "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
                       Section 6.2.1, IsRouter";
        }
        leaf mtu {
          type uint32 {
            range "1280..max";
          }
          units "octets";
          status deprecated;
          description
            "The size, in octets, of the largest IPv6 packet that the
             interface will send and receive.";
          reference
            "RFC 8200: Internet Protocol, Version 6 (IPv6)
                       Specification
                       Section 5";
        }
```

```
        list address {
          key "ip";
          status deprecated;
          description
            "The list of IPv6 addresses on the interface.";

          leaf ip {
            type inet:ipv6-address-no-zone;
            status deprecated;
            description
              "The IPv6 address on the interface.";
          }
          leaf prefix-length {
            type uint8 {
              range "0..128";
            }
            mandatory true;
            status deprecated;
            description
              "The length of the subnet prefix.";
          }
          leaf origin {
            type ip-address-origin;
            status deprecated;
            description
              "The origin of this address.";
          }
          leaf status {
            type enumeration {
              enum preferred {
                description
                  "This is a valid address that can appear as the
                   destination or source address of a packet.";
              }
              enum deprecated {
                description
                  "This is a valid but deprecated address that should
                   no longer be used as a source address in new
                   communications, but packets addressed to such an
                   address are processed as expected.";
              }
              enum invalid {
                description
                  "This isn't a valid address, and it shouldn't appear
                   as the destination or source address of a packet.";
              }
```

```
            enum inaccessible {
              description
                "The address is not accessible because the interface
                 to which this address is assigned is not
                 operational.";
            }
            enum unknown {
              description
                "The status cannot be determined for some reason.";
            }
            enum tentative {
              description
                "The uniqueness of the address on the link is being
                 verified.  Addresses in this state should not be
                 used for general communication and should only be
                 used to determine the uniqueness of the address.";
            }
            enum duplicate {
              description
                "The address has been determined to be non-unique on
                 the link and so must not be used.";
            }
            enum optimistic {
              description
                "The address is available for use, subject to
                 restrictions, while its uniqueness on a link is
                 being verified.";
            }
          }
          status deprecated;
          description
            "The status of an address.  Most of the states correspond
             to states from the IPv6 Stateless Address
             Autoconfiguration protocol.";
          reference
            "RFC 4293: Management Information Base for the
                       Internet Protocol (IP)
                       - IpAddressStatusTC
             RFC 4862: IPv6 Stateless Address Autoconfiguration";
        }
      }
```

```
      list neighbor {
        key "ip";
        status deprecated;
        description
          "A list of mappings from IPv6 addresses to
           link-layer addresses.

           This list represents the Neighbor Cache.";
        reference
          "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)";

        leaf ip {
          type inet:ipv6-address-no-zone;
          status deprecated;
          description
            "The IPv6 address of the neighbor node.";
        }
        leaf link-layer-address {
          type yang:phys-address;
          status deprecated;
          description
            "The link-layer address of the neighbor node.";
        }
        leaf origin {
          type neighbor-origin;
          status deprecated;
          description
            "The origin of this neighbor entry.";
        }
        leaf is-router {
          type empty;
          status deprecated;
          description
            "Indicates that the neighbor node acts as a router.";
        }
        leaf state {
          type enumeration {
            enum incomplete {
              description
                "Address resolution is in progress, and the
                 link-layer address of the neighbor has not yet been
                 determined.";
            }
            enum reachable {
              description
                "Roughly speaking, the neighbor is known to have been
                 reachable recently (within tens of seconds ago).";
            }
```

```
              enum stale {
                description
                  "The neighbor is no longer known to be reachable, but
                   until traffic is sent to the neighbor no attempt
                   should be made to verify its reachability.";
              }
              enum delay {
                description
                  "The neighbor is no longer known to be reachable, and
                   traffic has recently been sent to the neighbor.
                   Rather than probe the neighbor immediately, however,
                   delay sending probes for a short while in order to
                   give upper-layer protocols a chance to provide
                   reachability confirmation.";
              }
              enum probe {
                description
                  "The neighbor is no longer known to be reachable, and
                   unicast Neighbor Solicitation probes are being sent
                   to verify reachability.";
              }
            }
            status deprecated;
            description
              "The Neighbor Unreachability Detection state of this
               entry.";
            reference
              "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)
                         Section 7.3.2";
          }
        }
      }
    }
  }
  <CODE ENDS>
```

## 5. IANA Considerations

This document registers a URI in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registration has been made.

```
URI: urn:ietf:params:xml:ns:yang:ietf-ip
Registrant Contact: The NETMOD WG of the IETF.
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-ip
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ip
Prefix:       ip
Reference:    RFC 8344
```

## 6. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

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

ipv4/enabled and ipv6/enabled: These leafs are used to enable or disable IPv4 and IPv6 on a specific interface. By enabling a protocol on an interface, an attacker might be able to create an unsecured path into a node (or through it if routing is also enabled). By disabling a protocol on an interface, an attacker

might be able to force packets to be routed through some other interface or deny access to some or all of the network via that protocol.

ipv4/address and ipv6/address: These lists specify the configured IP addresses on an interface. By modifying this information, an attacker can cause a node to either ignore messages destined to it or accept (at least at the IP layer) messages it would otherwise ignore. The use of filtering or security associations may reduce the potential damage in the latter case.

ipv4/forwarding and ipv6/forwarding: These leafs allow a client to enable or disable the forwarding functions on the entity. By disabling the forwarding functions, an attacker would possibly be able to deny service to users. By enabling the forwarding functions, an attacker could open a conduit into an area. This might result in the area providing transit for packets it shouldn't, or it might allow the attacker access to the area, bypassing security safeguards.

ipv6/autoconf: The leafs in this branch control the autoconfiguration of IPv6 addresses and, in particular, whether or not temporary addresses are used. By modifying the corresponding leafs, an attacker might impact the addresses used by a node and -- thus, indirectly -- the privacy of the users using the node.

ipv4/mtu and ipv6/mtu: Setting these leafs to very small values can be used to slow down interfaces.

# 7. References

## 7.1. Normative References

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[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>.

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

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <https://www.rfc-editor.org/info/rfc4861>.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <https://www.rfc-editor.org/info/rfc4862>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <https://www.rfc-editor.org/info/rfc4941>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

[W3C.REC-xml-20081126] Bray, T., Paoli, J., Sperberg-McQueen, M., Maler, E., and F. Yergeau, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <https://www.w3.org/TR/2008/REC-xml-20081126>.

## 7.2. Informative References

[RFC826] Plummer, D., "An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <https://www.rfc-editor.org/info/rfc826>.

[RFC4293] Routhier, S., Ed., "Management Information Base for the Internet Protocol (IP)", RFC 4293, DOI 10.17487/RFC4293, April 2006, <https://www.rfc-editor.org/info/rfc4293>.

[RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <https://www.rfc-editor.org/info/rfc7217>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

[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>.

## Appendix A. Example: NETCONF <get-config> Reply

This section gives an example of a reply to the NETCONF <get-config> request for the running configuration datastore for a device that implements the data model defined in this document.

The XML [W3C.REC-xml-20081126] snippets that follow in this section and in Appendix B are provided as examples only.

```
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="101">
  <data>
    <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
      <interface>
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <mtu>1280</mtu>
          <address>
            <ip>2001:db8::10</ip>
            <prefix-length>32</prefix-length>
          </address>
          <dup-addr-detect-transmits>0</dup-addr-detect-transmits>
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

## Appendix B. Example: NETCONF <get-data> Reply

This section gives an example of a reply to the NETCONF <get-data> request for the operational state datastore for a device that implements the data model defined in this document.

This example uses the "origin" annotation, which is defined in the module "ietf-origin" [RFC8342].

```
<rpc-reply
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="101">
  <data xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-datastores">
    <interfaces
        xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
        xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type"
        xmlns:or="urn:ietf:params:xml:ns:yang:ietf-origin">

      <interface or:origin="or:intended">
        <name>eth0</name>
        <type>ianaift:ethernetCsmacd</type>
        <!-- other parameters from ietf-interfaces omitted -->

        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <enabled or:origin="or:default">true</enabled>
          <forwarding or:origin="or:default">false</forwarding>
          <mtu or:origin="or:system">1500</mtu>
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
            <origin>static</origin>
          </address>
          <neighbor or:origin="or:learned">
            <ip>192.0.2.2</ip>
            <link-layer-address>
              00:00:5E:00:53:AB
            </link-layer-address>
          </neighbor>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <enabled or:origin="or:default">true</enabled>
          <forwarding or:origin="or:default">false</forwarding>
          <mtu>1280</mtu>
```

```
            <address>
              <ip>2001:db8::10</ip>
              <prefix-length>32</prefix-length>
              <origin>static</origin>
              <status>preferred</status>
            </address>
            <address or:origin="or:learned">
              <ip>2001:db8::1:100</ip>
              <prefix-length>32</prefix-length>
              <origin>dhcp</origin>
              <status>preferred</status>
            </address>
            <dup-addr-detect-transmits>0</dup-addr-detect-transmits>
            <neighbor or:origin="or:learned">
              <ip>2001:db8::1</ip>
              <link-layer-address>
                00:00:5E:00:53:AB
              </link-layer-address>
              <origin>dynamic</origin>
              <is-router/>
              <state>reachable</state>
            </neighbor>
            <neighbor or:origin="or:learned">
              <ip>2001:db8::4</ip>
              <origin>dynamic</origin>
              <state>incomplete</state>
            </neighbor>
          </ipv6>
        </interface>
      </interfaces>
    </data>
  </rpc-reply>
```

## Acknowledgments

The author wishes to thank Jeffrey Lange, Ladislav Lhotka, Juergen Schoenwaelder, and Dave Thaler for their helpful comments.

## Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com