Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2014

M. Bjorklund
Tail-f Systems

October 18, 2013

A YANG Data Model for IP Management
draft-ietf-netmod-ip-cfg-11

Abstract

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

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 April 21, 2014.

Copyright Notice

Copyright (c) 2013 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

This document defines a YANG [RFC6020] data model for management of IP implementations.

The data model covers configuration of per-interface IPv4 and IPv6 parameters, and 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.

Parameters to manage IP routing are defined in [I-D.ietf-netmod-routing-cfg].

1.1. Terminology

The keywords "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].

The following terms are defined in [RFC6241] and are not redefined here:

- client
- server

The following terms are defined in [RFC6020] and are not redefined here:

- augment
- data model
- data node

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write) and "ro" state data (read-only).

- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. IP Data Model

This document defines the YANG module "ietf-ip", which augments the "interface" and "interface-state" lists defined in the "ietf-interfaces" module [I-D.ietf-netmod-interfaces-cfg] with IP specific nodes, and adds IP specific state data.

The data model has the following structure for IP configuration per interface:

```
   +--rw if:interfaces
      +--rw if:interface* [name]
         ...
         +--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 ip:prefix-length?   uint8
         |  |     +--:(netmask)
         |  |        +--rw ip:netmask?         yang:dotted-quad
         |  +--rw neighbor* [ip]
         |     +--rw ip                    inet:ipv4-address-no-zone
         |     +--rw link-layer-address    yang:phys-address
         +--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
            +--rw neighbor* [ip]
            |  +--rw ip                    inet:ipv6-address-no-zone
            |  +--rw link-layer-address    yang:phys-address
            +--rw dup-addr-detect-transmits?   uint32
            +--rw autoconf
               +--rw create-global-addresses?        boolean
               +--rw create-temporary-addresses?     boolean
               +--rw temporary-valid-lifetime?       uint32
               +--rw temporary-preferred-lifetime?   uint32
```

The data model defines two configuration containers per interface, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "enabled" that controls if the address family is enabled on that interface, and a leaf "forwarding"

that controls if IP packet forwarding for the address family is enabled on the interface. In each container, there is also a list of configured addresses, and a list of configured mappings from IP addresses to link-layer addresses.

The data model has the following structure for IP state per interface:

```
   +--ro if:interfaces-state
      +--ro if:interface* [name]
         ...
         +--ro ipv4!
         |  +--ro forwarding?   boolean
         |  +--ro mtu?          uint16
         |  +--ro address* [ip]
         |  |  +--ro ip               inet:ipv4-address-no-zone
         |  |  +--ro (subnet)?
         |  |  |  +--:(prefix-length)
         |  |  |  |  +--ro prefix-length?   uint8
         |  |  |  +--:(netmask)
         |  |  |     +--ro netmask?         yang:dotted-quad
         |  |  +--ro origin?          ip-address-origin
         |  +--ro neighbor* [ip]
         |     +--ro ip                    inet:ipv4-address-no-zone
         |     +--ro link-layer-address?   yang:phys-address
         |     +--ro origin?               neighbor-origin
         +--ro ipv6!
            +--ro forwarding?   boolean
            +--ro mtu?          uint32
            +--ro address* [ip]
            |  +--ro ip               inet:ipv6-address-no-zone
            |  +--ro prefix-length    uint8
            |  +--ro origin?          ip-address-origin
            |  +--ro status?          enumeration
            +--ro neighbor* [ip]
               +--ro ip                    inet:ipv6-address-no-zone
               +--ro link-layer-address?   yang:phys-address
               +--ro origin?               neighbor-origin
               +--ro is-router?            empty
               +--ro state?                enumeration
```

The data model defines two state containers per interface, "ipv4" and "ipv6", representing the IPv4 and IPv6 address families. In each container, there is a leaf "forwarding" that indicates if IP packet forwarding is enabled on that interface. In each container there is also a list of all addresses in use, and a list of known mappings from IP addresses to link-layer addresses.

3. Relationship to IP-MIB

If the device implements 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. The corresponding YANG data nodes are defined in [I-D.ietf-netmod-routing-cfg].

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

The following tables list the YANG data nodes with corresponding objects in the IP-MIB.

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

YANG interface configuration data nodes and related IP-MIB objects

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

YANG interface state data nodes and related IP-MIB objects

4. IP management YANG Module

This module imports typedefs from [RFC6991] and [I-D.ietf-netmod-interfaces-cfg], and references [RFC0791], [RFC0826], [RFC2460], [RFC4861], [RFC4862], and [RFC4941].

RFC Ed.: update the date below with the date of RFC publication and remove this note.

<CODE BEGINS> file "ietf-ip@2013-10-18.yang"

```
module ietf-ip {

  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 (NETCONF Data Modeling Language) Working Group";

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

     WG Chair: David Kessens
               <mailto:david.kessens@nsn.com>

     WG Chair: Juergen Schoenwaelder
               <mailto:j.schoenwaelder@jacobs-university.de>

     Editor:   Martin Bjorklund
               <mailto:mbj@tail-f.com>";

  description
    "This module contains a collection of YANG definitions for
     configuring IP implementations.

     Copyright (c) 2013 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.";

    // RFC Ed.: replace XXXX with actual RFC number and remove this
    // note.

    // RFC Ed.: update the date below with the date of RFC publication
    // and remove this note.
    revision 2013-10-18 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: 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 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
             auto-configuration.";
        }
        enum random {
          description
            "Indicates an address chosen by the system at
             random, e.g., an IPv4 address within 169.254/16, or an
             RFC 4941 privacy address.";
        }
      }
      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 e.g., IPv4 ARP or the IPv6 Neighbor Discovery
             protocol.";
        }
```

```
      }
      description
        "The origin of a neighbor entry.";
    }

    /*
     * Configuration data nodes
     */

    augment "/if:interfaces/if:interface" {
      description
        "Parameters for configuring IP 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 if IPv4 is enabled or disabled on this
             interface.";
        }
        leaf forwarding {
          type boolean;
          default false;
          description
            "Controls if IPv4 packet forwarding is enabled or disabled
             on this interface.";
        }
        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 configured 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.";
            }
          }
        }
        list neighbor {
          key "ip";
          description
            "A list of mappings from IPv4 addresses to
             link-layer addresses.

             Entries in this list are used as static entries in 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.";
          }
        }

      }
      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 if IPv6 is enabled or disabled on this
             interface.";
        }
        leaf forwarding {
          type boolean;
          default false;
          description
            "Controls if 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;
          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 2460: IPv6 Specification
                        Section 5";
         }
         list address {
           key "ip";
           description
             "The list of configured 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.";
           }
         }
         list neighbor {
           key "ip";
           description
             "A list of mappings from IPv6 addresses to
              link-layer addresses.

              Entries in this list are used as static entries in 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.";
           }
         }
```

```
        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 section 5.5 of RFC 4862.";
            reference
              "RFC 4862: IPv6 Stateless Address Autoconfiguration";
          }
          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";
          }
        }
      }
    }

    /*
     * Operational state data nodes
     */

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

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

        leaf forwarding {
          type boolean;
          description
            "Indicates if IPv4 packet forwarding is enabled or disabled
             on this interface.";
        }
        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.";
          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 {
            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;
            description
              "The origin of this address.";
          }
        }
        list neighbor {
          key "ip";
          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;
            description
              "The IPv4 address of the neighbor node.";
          }
          leaf link-layer-address {
            type yang:phys-address;
            description
              "The link-layer address of the neighbor node.";
          }
          leaf origin {
            type neighbor-origin;
            description
              "The origin of this neighbor entry.";
          }
        }

      }

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

        leaf forwarding {
          type boolean;
          default false;
          description
            "Indicates if 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;
          description
            "The size, in octets, of the largest IPv6 packet that the
             interface will send and receive.";
          reference
            "RFC 2460: 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;
            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.";
            }
          }
          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.

           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;
            description
              "The link-layer address of the neighbor node.";
          }
          leaf origin {
            type neighbor-origin;
            description
              "The origin of this neighbor entry.";
          }
          leaf is-router {
            type empty;
            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.";
                }
              }
              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 is requested to be 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 XXXX
```

6. Security Considerations

The YANG module defined in this memo is designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

There are a number of data nodes defined in the YANG module which 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 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 temporary addresses are used or not. 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. Acknowledgments

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

8. References

8.1. Normative References

[I-D.ietf-netmod-interfaces-cfg]
Bjorklund, M., "A YANG Data Model for Interface Configuration", draft-ietf-netmod-interfaces-cfg-12 (work in progress), July 2012.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, September 2007.

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

8.2. Informative References

[I-D.ietf-netmod-routing-cfg]
Lhotka, L., "A YANG Data Model for Routing Configuration", draft-ietf-netmod-routing-cfg-10 (work in progress), July 2012.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet

address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.

[RFC4293] Routhier, S., "Management Information Base for the Internet Protocol (IP)", RFC 4293, April 2006.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, March 2012.

Appendix A. Example: NETCONF <get> reply

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

```
<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">
      <interface>
        <name>eth0</name>
        <type>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>
    <interfaces-state
         xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type>ethernetCsmacd</type>
        <!-- other parameters from ietf-interfaces omitted -->

        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>false</forwarding>
          <mtu>1500</mtu>
          <address>
            <ip>192.0.2.1</ip>
            <prefix-length>24</prefix-length>
            <origin>static</origin>
          </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <forwarding>false</forwarding>
```

```
            <mtu>1500</mtu>
            <address>
              <ip>2001:db8::10</ip>
              <prefix-length>32</prefix-length>
              <origin>static</origin>
              <status>preferred</status>
            </address>
            <address>
              <ip>2001:db8::1:100</ip>
              <prefix-length>32</prefix-length>
              <origin>dhcp</origin>
              <status>preferred</status>
            </address>
            <neighbor>
              <ip>2001:db8::1</ip>
              <link-layer-address>00:01:02:03:04:05</link-layer-address>
              <origin>dynamic</origin>
              <is-router/>
              <state>reachable</state>
            </neighbor>
            <neighbor>
              <ip>2001:db8::4</ip>
              <origin>dynamic</origin>
              <state>incomplete</state>
            </neighbor>
          </ipv6>
        </interface>
      </interfaces-state>
    </data>
  </rpc-reply>
```

Author's Address

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com