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

M. Bjorklund
Tail-f Systems

March 2018

**Модель данных YANG для управления IP**

**A YANG Data Model for IP Management**

**Тезисы**

Этот документ определяет модель данных YANG для управления реализациями IP. Модель включает данные конфигурации и состояния системы.

Модель данных YANG в этом документе соответствует архитектуре хранилища данных сетевого управления NMDA[1], определенной в RFC 8342.

Документ отменяет действие RFC 7277.

**Статус документа**

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[2] и представляет согласованный взгляд сообщества IETF. Документ прошел открытое обсуждение и был одобрен для публикации IESG[3]. Не все одобренные IESG документы претендуют на статус Internet Standard (см. раздел 2 в RFC 7841).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc8344.



# Оглавление



## *1. Введение*

Этот документ определяет модель данных YANG [RFC7950] для управления реализациями IP.

Модель данных охватывает конфигурацию параметров IPv4 и IPv6 на уровне интерфейсов, а также отображение адресов IP на адреса канального уровня. Модель также дает информацию об используемых в работе адресах IP и имеющихся отображениях канального уровня. Параметры на уровне интерфейсов добавлены путем дополнения (augmentation) модели данных интерфейса, определенной в [RFC8343].

Эта версия модели данных IP поддерживает архитектуру хранилищ NMDA [RFC8342].

[1]Network Management Datastore Architecture.
[2]Internet Engineering Task Force.
[3]Internet Engineering Steering Group.

## 1.1. Отличия от RFC 7277

Ветви (subtree) ipv4 и ipv6 с узлами данных config false в ветви /interfaces-state/interface отменены. Все узлы данных config false сейчас присутствуют в ветвях ipv4 и ipv6 субдерева /interfaces/interface.

Серверы, не поддерживающие NMDA или желающие работать с клиентами, не поддерживающими NMDA, **могут** реализовать отмененные ветви ipv4 и ipv6 в субдереве /interfaces-state/interface.

## 1.2. Терминология

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

Перечисленные ниже термины определены в [RFC8342] и не определяются здесь заново.

- client (клиент);
- server (сервер);
- configuration (конфигурация);
- system state (состояние системы);
- intended configuration (предполагаемая конфигурация);
- running configuration datastore (хранилище данных рабочей конфигурации);
- operational state (операционное состояние);
- operational state datastore (хранилище данных операционного состояния).

Перечисленные ниже термины определены в [RFC7950] и не определяются здесь заново.

- augment (дополнение, усиление);
- data model (модель данных);
- data node (узел данных).

Терминология для описания моделей YANG представлена в [RFC7950].

## 1.3. Диаграммы деревьев

Диаграммы деревьев, используемыей в этом документе, следуют обозначениям, определенным в [RFC8340].

## 2. Модель данных IP

Этот документ определяет модуль YANG ietf-ip, который дополняет списки interface, определенные в модуле ietf-interfaces [RFC8343], связанными с IP узлами данных.

Структура узлов данных IP на уровне интерфейса, за исключением отмененных полей данных, приведена ниже.

```
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}?
```

Модель определяет для интерфейса два контейнера - ipv4 и ipv6, представляющие семейства адресов IPv4 и IPv6. В каждом контейнере имеется лист enabled, который показывает разрешено ли семейство адресов на данном интерфейсе, и лист forwarding, показывающий разрешена ли пересылка пакетов IP данного семейства на этом интерфейсе. В каждом контейнере имеется также список адресов и список отображений адресов IP на адреса канального уровня.

## 3. Связь с IP-MIB

Если устройство реализует IP-MIB [RFC4293], каждая запись в списках ipv4/address и ipv6/address отобрахается в один из объектов ipAddressEntry, где ipAddressIfIndex указывает address в объеке для интерфейса.

IP-MIB определяет объекты для управления сообщениями IPv6 Router Advertisement. Соответствующие узла данных YANG определены в [RFC8022].

Записи ipv4/neighbor и ipv6/neighbor отображаются на ipNetToPhysicalTable.

Ниже приведена таблица соответствия узлов данных YANG объектам IP-MIB.

*Узлы данных YANG для интерфейса и связанные объекты IP-MIB.*

| ***Узел данных YANG в /if:interfaces/if:interface*** | ***Объект IP-MIB*** |
|---|---|
| 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 |

## *4. Модуль YANG для управления IP*

Этот модуль импортирует определения типов (typedef) из [RFC6991] и [RFC8343], а они ссылаются на [RFC791], [RFC826], [RFC4861], [RFC4862], [RFC4941], [RFC7217] и [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";
  }

  /*
   * Функции
   */

  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";
  }
```

```
    /*
     * Определения типов
     */

    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.";
    }
```

```
    /*
     * Узлы данных
     */

    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";
        }
      }
    }
  }

  /*
   * Унаследованные узлы данных операционных состояний
   */

  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. Взаимодействие с IANA

Этот документ регистрирует URI в реестре IETF XML Registry [RFC3688]. В соответствии с форматом RFC 3688 выполнены перечисленные ниже регистрации.

```
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.
```

Этот документ регистрирует модуль YANG в реестре YANG Module Names [RFC6020].

```
Name:         ietf-ip
Namespace:    urn:ietf:params:xml:ns:yang:ietf-ip
Prefix:       ip
Reference:    RFC 8344
```

## 6. Вопросы безопасности

Описанный в этом документе модуль YANG определяет схему для данных, доступ к которым выполняется с помощью протоколов сетевого управления типа NETCONF [RFC6241] или RESTCONF [RFC8040]. Нижним уровнем NETCONF является защищенный транспортный уровень с обязательной реализациией защищенного транспорта SSH[1] [RFC6242]. Нижним уровнем RESTCONF является HTTPS с обязательной для реализацией транспорта TLS [RFC5246].

[1]Secure Shell — защищенная среда выполнения команд.

Модель управления доступом NETCONF [RFC8341] обеспечивает меры по ограничению доступа для отдельных пользователей NETCONF или RESTCONF предопределенным подмножеством всех доступных операций и содержимого NETCONF или RESTCONF.

Многие модели данных, определенные в этом модуле YANG обеспечивают возможность записи, создания и удаления (т. е. по умолчанию установлено config true). Эти узлы данных могут содержать конфиденциальную (sensitive) информацию , а также быть уязвимыми в некоторых сетевых средах. Операции записи (например, edit-config) для таких узлов без подобающей защиты могут оказывать нежелательное воздействие на работу сети. Ниже перечислены ветви (subtree) и узлы данных, которые могут включать конфиденциальную информацию или уязвимости.

***ipv4/enabled и ipv6/enabled***

Эти листья используются для включения или отключения IPv4 и IPv6 на конкретных интерфейсах. Включив протокол на интерфейсе, атакующий может получить возможность создания незащищенного пути к узлу (или через него, если включена маршрутизация). Отключив протокол на интерфейсе, атакующий сможет вынудить к маршрутизации пакетов через другой интерфейс или заблокировать доступ к части или всей сети по этому протоколу.

***ipv4/address и ipv6/address***

Эти списки задают адреса IP, установленные на интерфейсе. Изменяя эту информацию, атакующий сможет заставить узел игнорировать направленные ему сообщения или воспринимать (по меньшей мере на уровне IP) сообщения, которое он иначе бы игнорировал. Использование фильтрации или защищенных связей может снизить уровень возможных повреждений в последнем случае.

***ipv4/forwarding и ipv6/forwarding***

Эти листья позволяют клиенту включать и выключать функции пересылки. Запретив пересылку, атакующий может потенциально заблокировать обслуживание пользователей. Включив функции пересылки, атакующий сможет открыть для себя проход, что может привести к нежелательному транзиту пакетов или проникновению злоумышленника в сеть в обход механизмов защиты.

***ipv6/autoconf***

Листья в этой ветви управляют автоматической настройкой адресов IPv6 и, в частности, возможностью использования временных адресов. Изменяя соответствующие листься, атакующий может влиять на адреса, применяемые узлом и, опосредованно, на приватность пользователей узла.

***ipv4/mtu и ipv6/mtu***

Установка для этих листьев очень малых значений может существенно замедлить работу интерфейсов.

# *7. Литература*

## 7.1. Нормативные документы


[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. Дополнительная литература

[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>.

# *Прилежение A. Пример отклика NETCONF <get-config>*

В этом приложении дан пример отклика на запрос NETCONF <get-config> для рабочего хранилища конфигурации на устройстве, которое реализует определенную в этом документе модель данных.

Фрагменты XML [W3C.REC-xml-20081126] в этом и следующем приложении даны лишь в качестве примеров.

```
<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>
```

## Приложение B. Пример отклика NETCONF <get-data>

В этом приложении дан пример отклика на запрос NETCONF <get-data> для хранилища состояния на устройстве, поддерживающем описанную в этом документе модель данных.

В примере используется аннотация origin, определенная в модуле 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>
```

## Благодарности

Автор благодарит Jeffrey Lange, Ladislav Lhotka, Juergen Schoenwaelder и Dave Thaler за их полезные комментарии.

## *Адрес автора*

**Martin Bjorklund**

Tail-f Systems

Email: mbj@tail-f.com

**Перевод на русский язык**

Николай Малых

nmalykh@gmail.com