Abstract

This document defines a YANG data model for DHCP Server, relay, and client, including configuration and running state.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 12, 2009.

Copyright Notice

Copyright (c) 2018 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
1. Introduction

This document defines a YANG [RFC6020] data model that can be used to configure and manage DHCP.

This data model includes configuration data and state data, in which DHCP server, relay, and client nodes are defined.

1.1. 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].

The following terms are used within this document:

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

- client
- configuration data
- server
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" means 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. Design of Data Model

The goal of this document is to define a data model that provides a common user interface to the DHCP protocol. There is very information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

2.1. Overview

The overall structure of the model is described as the following.

```
module: ietf-dhcp
    +--rw dhcp
        |    +--rw server
        |    |    ...
        |    |    ...
        +--rw relay
```

Furthermore, design three if-feature for deployment of DHCP server/relay/client as below:

```yaml
feature dhcp-server {
  description "Feature DHCP server";
}
feature dhcp-client {
  description "Feature DHCP client";
}
feature dhcp-relay {
  description "Feature DHCP relay";
}
```

### 2.2. DHCP Server

Server contain the configuration items, including lease time, ping packet, IP address pool, option. And also include the operational data for server.

The most important part is the IP pool configuration. Specifying IP address section is for dynamic allocation. Configuring the mapping between IP address and MAC address is for manual allocation.

```yaml
module: ietf-dhcp
+-rw server {server}?
  |   +-rw lease-time?      uint32
  |   +-rw ping-packet-number?  uint8
  |   +-rw ping-packet-timeout? uint16
  |   |   +-rw option
  |     |     |   +-rw dhcp-server-identifier?  inet:ip-address
  |     |     |   +-rw domain-name?    string
  |     |     |   +-rw domain-name-server?  inet:ip-address
  |     |     |   +-rw interface-mtu?   uint32
  |     |     |   +-rw netbios-name-server?  inet:ip-address
  |     |     |   +-rw netbios-node-type? uint32
  |     |     |   +-rw netbios-scope?  string
  |     |   +-rw ip-pool* [ip-pool-name]
  |     |     |   +-rw ip-pool-name      string
  |     |     |   +-rw interface?      if:interface-ref
  |     |     |   +-rw gateway-ip?     inet:ip-address
```
2.3. DHCP Relay

The relay function is configured per interface. Enable/disable relay functionality on a specific interface, and specify the DHCP server.
2.4. DHCP Client

DHCP client is also managed per interface, including enable/disable client DHCP client function, client id and lease time. The packet statistics for the DHCP client is also included.

module: ietf-dhcp
  +--rw dhcp
    +--rw server
      ...
      ...
    +--rw relay {relay}?
      +--rw server-group* [server-group-name]
        +--rw server-group-name string
        +--rw interface? if:interface-ref
        +--rw gateway-address? inet:ipv4-address
        +--rw server-address* inet:ipv4-address
        +--ro packet-statistics
          +---x clean-statistics
          +--ro receive
            +--ro offer-packet? uint32
            +--ro ack-packet? uint32
            +--ro nack-packet? uint32
            +--ro decline-packet? uint32
            +--ro discover-packet? uint32
            +--ro request-packet? uint32
            +--ro release-packet? uint32
            +--ro inform-packet? uint32
          +--ro send
            +--ro offer-packet? uint32
            +--ro ack-packet? uint32
            +--ro nack-packet? uint32
            +--ro decline-packet? uint32
            +--ro discover-packet? uint32
            +--ro request-packet? uint32
            +--ro release-packet? uint32
            +--ro inform-packet? uint32
DHCP YANG Module

<CODE BEGINS> file "ietf-dhcp@2108-10-11.yang"
module ietf-dhcp {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dhcp";
    prefix "dhcp";

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

    organization "IETF dhc (Dynamic Host Configuration Protocol) Working Group";
    contact "leo.liubing@huawei.com
        loukunkun@huawei.com
        chin.chen@ericsson.com";
    description "The module for implementing DHCP protocol";

    revision "2018-10-11" {
        description "initial draft revision";
    }
}

3. DHCP YANG Module

...
Internet-Draft          DHCP Yang Data Model              October 2018

/*--------------------*/
/*      Features      */
/*--------------------*/
feature server {
  description "Feature DHCP server";
}

feature client {
  description "Feature DHCP client";
}
feature relay {
  description "Feature DHCP relay";
}

/*--------------------*/
/* Types Definition */
/*--------------------*/
typedef allocate-type {
  type enumeration {
    enum automatic {
      description "DHCP assigns a permanent IP address to a client";
    }
    enum dynamic {
      description "DHCP assigns an IP address to a client for a limited period of time";
    }
    enum manual {
      description "a client’s IP address is assigned by the network administrator, and DHCP is used simply to convey the assigned address to the client";
    }
  }
  description "Mechanisms for IP address allocation";
}

/*--------------------*/
/* Groupings         */
/*--------------------*/
grouping server-packet {
  description "The packets are sent from server";
}
leaf offer-packet {
    type uint32;
    config "false";
    description "Total number of DHCP OFFER packets";
}

leaf ack-packet {
    type uint32;
    config "false";
    description "Total number of DHCP ACK packets";
}

leaf nack-packet {
    type uint32;
    config "false";
    description "Total number of DHCP NACK packets";
}

leaf decline-packet {
    type uint32;
    config "false";
    description "Total number of DHCP DECLINE packets";
}

leaf discover-packet {
    type uint32;
    config "false";
    description "Total number of DHCP DISCOVER packets";
}

leaf request-packet {
    type uint32;
    config "false";
    description "Total number of DHCP REQUEST packets";
}

leaf release-packet {
    type uint32;
    config "false";
    description "Total number of DHCP RELEASE packets";
}

leaf inform-packet {
    type uint32;
    config "false";
    description "Total number of DHCP INFORM packets";
}

grouping client-packet {
    description "The packets are sent from client ";
}
grouping sum-packet {
    description "All of communicated packets between server and client";
    uses server-packet;
    uses client-packet;
}

grouping dhcp-option {
    description "Configuration option";
    leaf dhcp-server-identifier {
        type inet:ip-address;
        description "DHCP server identifier";
    }
    leaf domain-name {
        type string;
        description "Name of the domain";
    }
    leaf domain-name-server {
        type inet:ip-address;
        description "IPv4 address of the domain";
    }
    leaf interface-mtu {
        type uint32 {
            range "0..65535";
        }
        description "Minimum Transmission Unit (MTU) of the interface";
    }
    leaf netbios-name-server {
        type inet:ip-address;
        description "NETBIOS name server";
    }
    leaf netbios-node-type {
        type uint32 {
            range "0..65535";
        }
        description "NETBIOS node type";
    }
    leaf netbios-scope {
        type string;
        description "NETBIOS scope";
    }
}

/*******************/
/* Configuration Data */
container dhcp {
  description  "DHCP configuration";
  container server {
    if-feature server;
    description  "DHCP server configuration";
    leaf lease-time {
      type uint32{
        range "180..31536000";
      }
      description  "Default network address lease time assigned to DHCP clients";
    }
    leaf ping-packet-number{
      type uint8 {
        range "0..10";
      }
      default "0";
      description  "Number of ping packets";
    }
    leaf ping-packet-timeout {
      type uint16 {
        range "0..10000";
      }
      default "500";
      description  "Timeout of ping packet";
    }
    container option {
      description  "Configuration option";
      uses dhcp-option;
    }
  }
}
list ip-pool {
  key "ip-pool-name";
  description  "Global IP pool configuration";
  leaf ip-pool-name {
    type string {
      length "1..64";
    }
    description  "Name of the IP pool";
  }
  leaf interface {
    type if:interface-ref;
    description  "Name of the interface";
  }

leaf gateway-ip {
    type inet:ip-address;
    description "IPv4 address of the gateway";
}

leaf gateway-mask {
    type inet:ip-prefix;
    description "Network submask of the gateway";
}

leaf lease-time {
    type uint32 {
        range "180..31536000";
    }
    description "Default network address lease time assigned to DHCP clients";
}

list manual-allocation {
    key "mac-address ip-address";
    description "Mapping from MAC address to IP address";

    leaf mac-address {
        type yang:mac-address;
        description "MAC address of the host";
    }

    leaf ip-address {
        type inet:ip-address;
        description "IPv4 address of the host";
    }
}

list section {
    key "start-ip";
    description "IPv4 address for the range";

    leaf start-ip {
        type inet:ipv4-address;
        mandatory "true";
        description "Starting IPv4 Address of a section";
    }

    leaf end-ip {
        type inet:ipv4-address;
        description "Last IPv4 Address of a section";
    }
}

container option {
    description "Configuration option";
    uses dhcp-option;
}
leaf used-ip-count {
  type uint32;
  config "false";
  description "Total number of used IPv4 addresses";
}
leaf idle-ip-count {
  type uint32;
  config "false";
  description "Total number of idle IPv4 addresses";
}
leaf conflict-ip-count {
  type uint32;
  config "false";
  description "Total number of conflict IPv4 addresses";
}
leaf total-ip-count {
  type uint32;
  config "false";
  description "Total number of IPv4 addresses";
}

list packet-statistics {
  key "interface";
  config "false";
  description "Packet statistics";
  leaf interface {
    type if:interface-ref;
    description "Name of the interface";
  }
  action clean-statistics {
    description "Clear the specific interface statistics";
  }
  container receive {
    description "Number of received packets";
    uses client-packet;
  }
  container send {
    description "Number of sent packets";
    uses server-packet;
  }
}
container host {
  config "false";
  description "Host status information";
  leaf interface {
    type if:interface-ref;
    description "Name of the interface";
  }
  leaf host-ip {
    type string;
    description "IPv4 address of the host";
  }
  leaf host-hardware-address {
    type string;
    description "MAC address of the host";
  }
  leaf lease {
    type uint32;
    description "Default network address lease time assigned to DHCP clients";
  }
  leaf type {
    type allocate-type;
    description "Mechanisms for IP address allocation";
  }
}

container relay {
  if-feature relay;
  description "DHCP relay agent configuration";
  list server-group {
    key "server-group-name";
    description "DHCP server group configuration that DHCP relays to";
    leaf server-group-name {
      type string;
      description "Name of a DHCP server group";
    }
    leaf interface {
      type if:interface-ref;
      description "Name of the interface";
    }
    leaf gateway-address {
      type inet:ipv4-address;
      description "IPv4 address of the gateway";
    }
    leaf-list server-address {
      type inet:ipv4-address;
      description "List of IPv4 addresses for DHCP relay";
    }
  }
}
type inet:ipv4-address;
  description "IPv4 address of the server";
}

container packet-statistics {
  config "false";
  description "Packet statistics";

  action clean-statistics {
    description "Clear the specific interface statistics";
  }

  container receive {
    description "Number of received packets";
    uses sum-packet;
  }

  container send {
    description "Number of sent packets";
    uses sum-packet;
  }
}

container client {
  if-feature client;
  description "DHCP client configuration";

  list interfaces {
    key "interface";
    description "Interface configuration";

    leaf interface {
      type if:interface-ref;
      description "Name of the interface";
    }

    leaf client-id {
      type string;
      description "DHCP client identifier";
    }

    leaf lease {
      type uint32 {
        range "1..4294967295";
      }
      description "Default network address lease time assigned to DHCP clients";
    }
}
container packet-statistics {
  config "false";
  description "Packet statistics";

  action clean-statistics {
    description "Clear the specific interface statistics";
  }

  container receive {
    description "Number of received packets";
    uses server-packet;
  }

  container send {
    description "Number of sent packets";
    uses client-packet;
  }
}

4. Security Considerations

The data model defined does not create any security implications.

5. Contributors

The following people all contributed significantly to the initial YANG model:

- Gang Yan (Huawei)
- Hongying Sheng (Ericsson)

6. IANA Considerations

This draft does not request any IANA action.
7. Normative References


Authors’ Addresses