Internet Engineering Task Force
Internet Draft
Intended status: Standards Track
Expires: September 2017

B. Liu, Ed.
K. Lou
Huawei Technologies
C. Chen
Ericsson
March 7, 2017

Yang Data Model for DHCP Protocol
draft-liu-dhc-dhcp-yang-model-06.txt

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 September 7, 2017.

Table of Contents

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, replay, 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

- state data

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

- augment
- data model
- data node
- presence container

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
```

```
      |  |   ...
      |  |   ...
      |  +--rw client
      |  |   ...
      |  |   ...
      +--ro dhcp-state
         +--ro server
         |  ...
         |  ...
         +--ro relay
         |  ...
         |  ...
         +--ro client
            ...
            ...
```

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

```
  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 configurations contain lease time, ping packet, IP address pool and option configuration.

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.

```
module: ietf-dhcp
   +--rw dchp
      +--rw 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 dhcp-ip-pool* [ip-pool-name]
               +--rw ip-pool-name         string
               +--rw interface?           if:interface-ref
               +--rw gateway-ip?          inet:ip-address
               +--rw gateway-mask?        inet:ip-prefix
               +--rw lease-time?          uint32
               +--rw manual-allocation* [mac-address ip-address]
               |  +--rw mac-address    yang:mac-address
               |  +--rw ip-address     inet:ip-address
               +--rw section* [section-index]
               |  +--rw section-index       uint16
               |  +--rw section-start-ip    inet:ipv4-address
               |  +--rw section-end-ip?     inet:ipv4-address
               +--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
```

2.3. DHCP Relay

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

```
module: ietf-dhcp
   +--rw dchp
      +--rw server
            ...
            ...
      +--rw 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
```

2.4. DHCP Client

DHCP client is also managed per interface, including enable/disable client DHCP client function, client id and lease time.

```
module: ietf-dhcp
   +--rw dchp
      +--rw server
            ...
            ...
      +--rw relay
            ...
            ...
      +--rw client
         +--rw interfaces* [interface]
            +--rw interface?   if:interface-ref
            +--rw client-id?   string
            +--rw lease?       uint32
```

2.5. DHCP State

The " dhcp-state" records the package statistic information and host allocated status, including server, relay and client.

```
module: ietf-dhcp
   +--rw dhcp
      +--rw server
            ...
            ...
      +--rw relay
            ...
            ...
      +--rw client
            ...
            ...
   +--ro dhcp-state
      +--ro server {server}?
      |  +--ro packet-statistics
      |  |  +--ro interface?   if:interface-state-ref
      |  |  +--ro receive
      |  |  |  +--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 host
          |  |  +--ro interface?               string
          |  |  +--ro host-ip?                 string
          |  |  +--ro host-hardware-address?   string
          |  |  +--ro lease?                   uint32
          |  |  +--ro type?                    allocate-type
          |  +--ro ip-pool* [ip-pool-name]
          |     +--ro ip-pool-name         string
          |     +--ro gateway-ip?          inet:ip-address
          |     +--ro gateway-mask?        inet:ip-prefix
          |     +--ro used-ip-count?       uint32
          |     +--ro idle-ip-count?       uint32
          |     +--ro conflict-ip-count?   uint32
          |     +--ro total-ip-count?      uint32
          +--ro relay {relay}?
          |  +--ro packet-statistics
          |     +--ro interface?   if:interface-state-ref
          |     +--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
          +--ro client {client}?
             +--ro packet-statistics
                +--ro interface?   if:interface-state-ref
                +--ro receive
                |  +--ro offer-packet?   uint32
                |  +--ro ack-packet?     uint32
                |  +--ro nack-packet?    uint32
                +--ro send
```

```
                +--ro decline-packet?    uint32
                +--ro discover-packet?   uint32
                +--ro request-packet?    uint32
                +--ro release-packet?    uint32
                +--ro inform-packet?     uint32
```

2.6. DHCP RPC

The " dhcp-rpc" is designed to clean the packet statistics on server/relay/client node.

```
rpcs:
   +---x clean-server-statistics     {dhcp-server}?
   |  +---w input
   |  |  +---w interface?   -> /dhcp-state/server/packet-statistics/
interface
   |  |  +---w clean-at?    yang:date-and-time
   |  +--ro output
   |     +--ro clean-finished-at?   yang:date-and-time
   +---x clean-relay-statistics      {dhcp-relay}?
   |  +---w input
   |  |  +---w interface?   -> /dhcp-state/relay/packet-statistics/interface
   |  |  +---w clean-at?    yang:date-and-time
   |  +--ro output
   |     +--ro clean-finished-at?   yang:date-and-time
   +---x clean-client-statistics     {dhcp-client}?
      +---w input
      |  +---w interface?   -> /dhcp-state/client/packet-statistics/
interface
      |  +---w clean-at?    yang:date-and-time
      +--ro output
         +--ro clean-finished-at?   yang:date-and-time
```

3. DHCP YANG Module

```
<CODE BEGINS> file "ietf-dhcp@2017-03-02.yang"
module ietf-dhcp {
  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 2017-03-02 {
  description   "initial draft revision";
  reference     "rfc2131 rfc6020";
}

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

feature dhcp-client {
  description "Feature DHCP client";
}

feature dhcp-relay {
  description "Feature DHCP relay";
}

/*--------------------*/
/*  Types Defination  */
/*--------------------*/
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 DHCPOFFER packets";
      }
      leaf ack-packet {
        type uint32;
        config "false";
        description "Total number of DHCPACK packets";
      }
      leaf nack-packet {
        type uint32;
        config "false";
        description "Total number of DHCPNAK packets";
      }
    }

    grouping client-packet {
      description "The packets are sent from client ";

      leaf decline-packet {
        type uint32;
        config "false";
        description "Total number of DHCPDECLINE packets";
      }
      leaf discover-packet {
        type uint32;
        config "false";
        description "Total number of DHCPDISCOVER packets";
      }
      leaf request-packet {
        type uint32;
        config "false";
```

```
        description "Total number of DHCPREQUEST packets";
      }
      leaf release-packet {
        type uint32;
        config "false";
        description "Total number of DHCPRELEASE packets";
      }
      leaf inform-packet {
        type uint32;
        config "false";
        description "Total number of DHCPINFORM packets";
      }
    }

    grouping sum-packet {
      description "All of commnicated 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 dhcp-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 dhcp-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 "section-index";
              description "IPv4 address for the range";
              leaf section-index {
```

```
            type uint16 {
                    range "0..255";
            }
            description "Index of IPv4 address range";
          }
          leaf section-start-ip {
            type inet:ipv4-address;
            mandatory "true";
            description "Starting IPv4 Address of a section";
          }
          leaf section-end-ip {
            type inet:ipv4-address;
            description "Last IPv4 Address of a section";
          }
        }
        container option {
          description "Configuration option";
          uses dhcp-option;
        }
      }
    }
    container relay {
      if-feature dhcp-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 "IPv4 address of the server";
        }
      }
    }
```

```
       container client {
         if-feature dhcp-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";
           }
         }
       }
     }

     /*--------------------------*/
     /*  Operational State Data  */
     /*--------------------------*/
     container dhcp-state {
       config "false";
       description "DHCP state data";

       container server {
         if-feature dhcp-server;
         description "DHCP server state data";

         container packet-statistics {
           description "Packet statistics";

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

           container receive {
             description "Number of  received packets";
```

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

            uses server-packet;
          }
        }
        container host {
          description "Host status information";
          leaf interface {
            type string;
            config "false";
            description "Name of the interface";
          }
          leaf host-ip {
            type string;
            config "false";
            description "IPv4 address of the host";
          }
          leaf host-hardware-address {
            type string;
            config "false";
            description "MAC address of the host";
          }
          leaf lease {
            type uint32;
            config "false";
            description "Default network address lease
                         time assigned to DHCP clients";
          }
          leaf type {
            type allocate-type;
            config "false";
            description "Mechanisms for IP address allocation";
          }
        }
        list ip-pool {
          key "ip-pool-name";
          description   "Global IP pool configuration";

          leaf ip-pool-name {
            type string {
              length "1..64";
            }
            description "Name of an IP pool";
```

```
          }
          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 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";
          }
        }
      }
      container relay {
        if-feature dhcp-relay;
        description "DHCP reply agent state data";

        container packet-statistics {
          description "Packet statistics";

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

          container receive {
            description "Number of  received packets";

            uses sum-packet;
```

```
          }
          container send {
            description
              "Number of sent packets";

            uses sum-packet;
          }
        }
      }
      container client {
        if-feature dhcp-client;
        description "DHCP client state data";

        container packet-statistics {
          description "Packet statistics";

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

          container receive {
            description "Number of  received packets";

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

            uses client-packet;
          }
        }
      }
    }

    /*--------------------------*/
    /*  Define RPC for action   */
    /*--------------------------*/
    rpc clean-server-statistics {
      if-feature dhcp-server;
      description "Clean server packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:server/"+
              "dhcp:packet-statistics/dhcp:interface";
          }
```

```
          description "Name of the interface";
        }
        leaf clean-at {
          type yang:date-and-time;
          description "The start time to clean packet statistics";
        }
      }
      output {
        leaf clean-finished-at {
          type yang:date-and-time;
          description "The finish time to clean packet statistics";
        }
      }
    }

    rpc clean-relay-statistics {
      if-feature dhcp-relay;
      description "Clean relay packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:relay/"+
              "dhcp:packet-statistics/dhcp:interface";
          }
          description "Name of the interface";
        }
        leaf clean-at {
          type yang:date-and-time;
          description "The start time to clean packet statistics";
        }

      }
      output {
        leaf clean-finished-at {
          type yang:date-and-time;
          description "The finish time to clean packet statistics";
        }
      }
    }

    rpc clean-client-statistics {
      if-feature dhcp-client;
      description "Clean client packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:client/"+
```

```
              "dhcp:packet-statistics/dhcp:interface";
            }
            description "Name of the interface";
          }
          leaf clean-at {
            type yang:date-and-time;
            description "The start time to clean packet statistics";
          }

        }
        output {
          leaf clean-finished-at {
            type yang:date-and-time;
            description "The finish time to clean packet statistics";
          }
        }
      }
    }
    <CODE ENDS>
```

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

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <http://www.rfc-editor.org/info/rfc6021>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, September 2016, <http://www.rfc-editor.org/info/rfc7950>.

Authors' Addresses

Bing Liu
Huawei Technologies
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: leo.liubing@huawei.com

Kunkun Lou
Huawei Technologies
Huawei Nanjing R&D Center
101 Software Avenue, Yuhua District, Nanjing, Jiangsu, 210012
P.R. China

Email: loukunkun@huawei.com

Chin Chen
Ericsson (China) Communications Company Ltd.
Ericsson Tower, No. 5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email: chin.chen@ericsson.com