A YANG Data Model for DHCP Configuration
draft-liu-dhc-dhcp-yang-model-00

Abstract

This document defines a YANG data model for configuring DHCP Server, relay, and client.

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 July 4, 2015.

Copyright Notice

Copyright (c) 2014 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.
1. Introduction

This document defines a YANG [RFC6020] [RFC6021] data model for configuring DHCP Server, relay, and client.

This model is constructed based on IPv4 version of DHCP[RFC2131]. This model contains three roles of a DHCP system: DHCP server, DHCP relay and DHCP client. A device could be one of the roles, or a combination of two or three roles. When a device is configured multiple roles, the roles are independent with each other. In other words, this model is only a container for the roles, there is no intrinsic relationship between the roles.

2. Requirements Language and 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 [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

Terminology:

- DHCP: Dynamic Host Configuration Protocol [RFC2131]

3. DHCP YANG Model Overview

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

module: ietf-dhcp
  +--dhcp
++--relay
  +--rw dhcpRelayIfCfgs
    +--rw dhcpRelayIfCfgs* [ifName]
      +--rw ifName                     string
      +--rw enable                     boolean
      +--rw serverGroupName            string
      +--rw serverAddress              inet:ipv4-address
  +--rw dhcpRelayServerGroups
    +--rw dhcpRelayServerGroup* [serverGroupName]
      +--rw serverGroupName        string
      +--rw vpnName                if:interface-ref
      +--rw sourceIP               inet:ip-address
      +--rw gateway                inet:ip-address
      +--rw serverAddress          enum
      +--r dhcpRelaySerGrpStats
        +--r dhcpRelaySerGrpStat* [serverIpAddr]
          +--r serverIpAddr                inet:ipv4-address
          +--r pktsReceiveFromClient       uint32
          +--r discoverPktReceive          uint32
          +--r requestPktsReceive          uint32
          +--r informPktsReceive           uint32
          +--r declinePktsReceive          uint32
          +--r pktsReceiveFromServers      uint32
          +--r offerPktsReceive            uint32
          +--r ackPktsReceive              uint32
          +--r nakPktsReceive              uint32
          +--r pktsSentToServers           uint32
          +--r pktsSentToClients           uint32
          +--r unicastPktsSentToClients    uint32
          +--r broadcastPktsSentToClients  uint32
    +--r dhcpRelayStatistics
      +--r badPacketsRecvd                uint32
      +--r packetsRecvdFromClient         uint32
      +--r discoverPacketsRecvd          uint32
      +--r requestPacketsRecvd           uint32
      +--r informPacketsRecvd            uint32
      +--r declinePacketsRecvd           uint32
      +--r packetsRecvdFromServers       uint32
      +--r offerPacketsRecvd             uint32
      +--r ackPacketsRecvd               uint32
      +--r nakPacketsRecvd               uint32
      +--r packetsSentToServers          uint32
      +--r packetsSentToClients          uint32
      +--r unicastPacketsSentToClients   uint32
      +--r broadcastPacketsSentToClients uint32
      +--r releasePacketsSentToServers   uint32
---r sendRenewPacket  uint32

---server
    ++-rw common
        ++-rw pingPacketNumber  uint8
        ++-rw pingPacketTimeOut  uint16
    ++-rw globalIpPools
        ++-rw globalIpPool*  [ipPoolName]
            ++-rw ipPoolName  string
            ++-rw vpnInstance  string
            ++-rw gatewayIp  inet:ipv4-address
            ++-rw gatewayMask  inet:ipv4-address
        ++-rw sections
            ++-rw section*  [sectionIndex]
                ++-rw sectionIndex  uint16
                ++-rw sectionStartIp  inet:ipv4-address
                ++-rw sectionEndIp  inet:ipv4-address
                ++-r ipPoolSectionStat
                    ++-r usedIpCount  uint32
                    ++-r idleIpCount  uint32
                    ++-r conflictIpCount  uint32
                    ++-r totalIpCount  uint32
            ++-rw leaseTime
                ++-rw day  uint16
                ++-rw hour  uint8
                ++-rw minute  uint8
            ++-rw domainNameServer  inet:ipv4-address
            ++-rw domainName  string
            ++-rw NbnsServer  inet:ipv4-address
            ++-rw NbNodeType  enum
            ++-r UserDefOptions
                ++-r UserDefOptions*  [optionCode]
                    ++-rw optionCode  uint8
                    ++-rw ipAddress  inet:ipv4-address
                    ++-rw optionString  string
                    ++-rw optionHex  string
                ++-r ipPoolStat
                    ++-r usedIpCount  uint32
                    ++-r idleIpCount  uint32
                    ++-r conflictIpCount  uint32
                    ++-r totalIpCount  uint32
            ++-r packetStatistics
                ++-r clientRequestCount  uint32
                ++-r discoverCount  uint32
                ++-r requestCount  uint32
                ++-r declineCount  uint32
                ++-r releaseCount  uint32
3.1. DHCP Relay

The relay function is configured in a per interface manner. Thus, there is a "dhcpRelayIfCfgs" container to list each interface’s general relay configurations, which mainly include enable/disable of relay, server address, and server group (see below).

In some scenarios, there are multiple DHCP servers for high reliable, load balancing or other considerations. The servers could combined as multiple groups, and each group is binding to a specific relay configuration (as decribed in above "dhcpRelayIfCfgs"). The groups are listed in the "dhcpRelayServerGroups" container. The "dhcpRelaySerGrpStats" container records statistic information by each DHCP server IP address in the group.

The "dhcpRelayStatistics" list records the statistic information of the whole relay entity.

3.2. DHCP Server

Server configurations contain common configurations, IP address pool configuration and statistic information.
Each time the DHCP server intends to allocate an IP address, it needs to confirm whether the address has been occupied or not through pinging. The "common" container includes two parameters to control the packet number and timeout period respectively.

The most important part of server configurations is the IP pool configuration. Normally, the DNS configuration and some other option configuration are relevant to the IP pool where the allocated IP address comes from, so the option configuration is sorted under the IP pool container. This model supports user defined options configuration through the "UserDefOptions" container.

3.3. DHCP Client

DHCP client is also managed in a per interface manner. Except for enable/disable of client function, other objects are all status information.

4. DHCP YANG Module

<CODE BEGINS>
module huawei-dhcp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-dhcp";
    prefix "dhcp";

    import ietf-inet-types {
        prefix "inet";
    }

    contact "leo.liubing@huawei.com" "loukunkun@huawei.com";
    description "The module for implementing DHCP protocol";
    revision "2014-12-18";

    container relay {
        container dhcpRelayIfCfgs {
            list dhcpRelayIfCfg {
                key "ifName";

                leaf ifName {
                    description "Specify the interface name that dhcp relay configured on";
                    type "string";
                    config "true";
                }
            }
        }
    }
}</CODE BEGINS>
leaf enable {
    description "Enable or disable dhcp relay function";
    type "boolean";
    default "false";
    config "true";
}

leaf serverGroupName {
    description "Server Group Name";
    type string;
    config "true";
}

leaf-list serverAddress {
    description "DHCP relay destination server IP address";
    type inet:ipv4-address;
    config "true";
}

container dhcpRelayServerGroups {
    list dhcpRelayServerGroup {
        key "serverGroupName";
        description "DHCP relay server group ";
        leaf serverGroupName {
            description "name of server group";
            type string;
            config "true";
        }
        leaf vpnName {
            description "VPN name for server group";
            type string;
            config "true";
        }
        leaf gateway {
            description "gateway for server group";
            type inet:ipv4-address;
            config "true";
        }
    }
}
leaf-list serverAddress {
    description "DHCP relay destination server IP address";
    type inet:ipv4-address;
    config "true";
}

container dhcpRelaySerGrpStats {
    list dhcpRelaySerGrpStats {
        description "DHCP relay server group packet statistics ";
        key "serverIpAddr";
        leaf serverIpAddr {
            type inet:ipv4-address;
        }
        leaf pktsReceiveFromClient {
            type "uint32";
            config "false";
        }
        leaf discoverPktsReceive {
            type "uint32";
            config "false";
        }
        leaf requestPktsReceive {
            type "uint32";
            config "false";
        }
        leaf releasePktsReceive {
            type "uint32";
            config "false";
        }
        leaf informPktsReceive {
            type "uint32";
            config "false";
        }
        leaf declinePktsReceive {
            type "uint32";
            config "false";
        }
        leaf pktsReceiveFromServers {
            type "uint32";
            config "false";
        }
        leaf offerPktsReceive {
            type "uint32";
            config "false";
        }
        leaf ackPktsReceive {
container dhcpRelayStatistics {
  leaf badPacketsRecvd {
    type "uint32";
    config "false";
  }
  leaf packetsRecvdFromClient {
    type "uint32";
    config "false";
  }
  leaf discoverPacketsRecvd {
    type "uint32";
    config "false";
  }
  leaf requestPacketsRecvd {
    type "uint32";
    config "false";
  }
}
leaf informPacketsRecvd {
    type "uint32";
    config "false";
}
leaf declinePacketsRecvd {
    type "uint32";
    config "false";
}
leaf releasePacketsRecvd {
    type "uint32";
    config "false";
}
leaf packetsRecvdFromServers {
    type "uint32";
    config "false";
}
leaf offerPacketsRecvd {
    type "uint32";
    config "false";
}
leaf ackPacketsRecvd {
    type "uint32";
    config "false";
}
leaf nakPacketsRecvd {
    type "uint32";
    config "false";
}
leaf packetsSentToServers {
    type "uint32";
    config "false";
}
leaf packetsSentToClients {
    type "uint32";
    config "false";
}
leaf unicastPacketsSentToClients {
    type "uint32";
    config "false";
}
leaf broadcastPacketsSentToClients {
    type "uint32";
    config "false";
}
leaf releasePacketsSentToServers {
    type "uint32";
    config "false";
leaf sendRenewPacket {
    type "uint32";
    config "false";
}
}

container server {
    container common {
        leaf pingPacketNumber {
            description "Ping packet number";
            type uint8 {
                range "0..10";
            }
            config "true";
            default "0";
        }
        leaf pingPacketTimeOut {
            description "Ping packet timeout";
            type uint16 {
                range "0..10000";
            }
            config "true";
            default "500";
        }
    }
    container globalIpPool {
        list globalIpPool {
            key "ipPoolName";
            leaf ipPoolName {
                description "IP pool name";
                type string {
                    length "1..64";
                }
                config "true";
            }
            leaf vpnInstance {
                description "VPN name";
                type string {
                    length "1..31";
                }
            }
        }
    }
}

config "true";
}
}

container gatewayIp {
  leaf gatewayIp {
    description "Gateway IP Address";
    type inet:ipv4-address;
    config "true";
  }
  leaf gatewayMask {
    description "Gateway IP Address Mask";
    type inet:ipv4-address;
    config "true";
  }
}
}

container sections {
  list section {
    key "sectionIndex";
    leaf sectionIndex {
      description "Section Index";
      type uint16 {
        range "0..255";
        config "true";
      }
    }
    leaf sectionStartIp {
      description "IP Address";
      type inet:ipv4-address;
      config "true";
      mandatory "true";
    }
    leaf sectionEndIp {
      description "IP Address";
      type inet:ipv4-address;
      config "true";
      type inet:ipv4-address;
      mandatory "false";
    }
  }
}

container ipPoolSectionStat {
  leaf usedIpCount {
    description "Used Ip Count";
  }
}
leaf idleIpCount {
    description "Idle Ip Count";
    type uint32;
    config "false";
}

leaf conflictIpCount {
    description "Conflict Ip Count";
    type uint32;
    config "false";
}

leaf totalIpCount {
    description "Total Ip Count";
    type uint32;
    config "false";
}

container leaseTime {
    description "Specifies the lease time, option code is 51.";
    leaf day {
        description "Day";
        type uint16 {
            range "0..49710";
        }
        config "true";
        default "1";
    }
    leaf hour {
        description "Hour";
        type uint8 {
            range "0..23";
        }
        config "true";
        default "0";
    }
    leaf minute {
        description "Minute";
        type uint8 {
            range "0..59";
        }
        config "true";
    }
}
leaf-list domainNameServer {
  description "Specifies the domain name server, option code is 5.";
  type inet:ipv4-address;
  config "true";
}

leaf domainName {
  description "Specifies the domain name, option code is 15.";
  type string {
    length "1..255";
  }
  config "true";
}

leaf-list NbnsServer {
  description "Specifies the NetBIOS name server, option code is 44.";
  type inet:ipv4-address;
  config "true";
}

leaf NbNodeType {
  description " Specifies the NetBIOS node type, option code is 46.";
  type enumeration {
    enum B-node {
      value "1";
    }
    enum P-node {
      value "2";
    }
    enum M-node {
      value "4";
    }
    enum H-node {
      value "8";
    }
  }
  config "true";
}

container UserDefOptions {
  list UserDefOption {
    description "Specifies the user defined DHCP options";
    key "optionCode";
  }
}
leaf optionCode {
  description "The option code of the user defined option";
  type uint8 {
    range "2 | 4..5 | 7..14 | 16..43 | 47..49 | 56 | 62..81 | 83..254";
    config "true";
  }
}

leaf-list ipAddress {
  description "IP address list";
  type inet:ipv4-address;
  config "true";
}

leaf optionString {
  description "User defined option string";
  type string {
    length "1..254";
  }
  config "true";
}

leaf optionHex {
  description "User defined option HEX string";
  type string {
    length "2..508";
  }
  config "true";
}

}

container ipPoolStat {

  leaf usedIpCount {
    description "Used Ip Count";
    type uint32;
    config "false";
  }

  leaf idleIpCount {
    description "Idle Ip Count";
    type uint32;
    config "false";
  }

  leaf conflictIpCount {
    description "Conflict Ip Count";
    type uint32;
  }

  }
container packetStatistics {

  leaf clientRequestCount {
    description "Client Request Count";
    type uint32;
    config "false";
  }

  leaf discoverCount {
    description "Discover Count";
    type uint32;
    config "false";
  }

  leaf requestCount {
    description "Request Count";
    type uint32;
    config "false";
  }

  leaf declineCount {
    description "Decline Count";
    type uint32;
    config "false";
  }

  leaf releaseCount {
    description "Release Count";
    type uint32;
    config "false";
  }

  leaf informCount {
    description "Inform Count";
    type uint32;
    config "false";
  }

  leaf serverReplyCount {
    description "Server Reply Count";
    type uint32;
  }

  leaf totalIpCount {
    description "Total Ip Count";
    type uint32;
    config "false";
  }
}

container client {

    container dhcpClientIf {

        list dhcpClientIf {
            key "ifName";

            leaf ifName {
                description "Specify the interface name that dhcp client configured on";
                type "string";
                config "true";
            }

            leaf enable {
                description "Enable or disable dhcp client function";
                type "boolean";
                default "false";
                config "true";
            }

            container dhcpClientStatus {

                description "Specify the status of DHCP client on the interface";

                leaf status {
                    description "Specify the status of DHCP client on the interface";
                }

            } // end dhcpClientStatus

        } // end dhcpClientIf

    } // end client

}
type enumeration {
    enum Init;
    enum Discoverying;
    enum Offered;
    enum Requesting;
    enum Acked;
}  
  config "false";
}

leaf clientIpAddr {
    description "Specify the IP address obtained from DHCP server on the interface";
    type inet:ipv4-address;
    config "false";
}

leaf-list dnsServerIpAddr {
    description "Specify the DNS server IP address obtained from DHCP server on the interface";
    type inet:ipv4-address;
    config "false";
}

container dhcpClientIfStatistics {
    description "Specify the statistics of DHCP client send or receive packets on the interface";
    leaf discoverCount {
        description "Discover Count";
        type uint32;
        config "false";
    }

    leaf requestCount {
        description "Request Count";
        type uint32;
        config "false";
    }

    leaf declineCount {
        description "Decline Count";
        type uint32;
        config "false";
    }

    leaf releaseCount {
        description "Release Count";
    }
}
5. Security Considerations

TBD.

6. IANA Considerations

TBD.
7. Acknowledgements

Valuable comment was received from Gang Yan and Guangying Zheng to improve the draft.

This document was produced using the xml2rfc tool [RFC2629].

8. Normative References


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