DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 25, 2015

Y. Cui
H. Wang
L. Sun
Tsinghua University
March 24, 2015

YANG Data Model for DHCPv6 Configuration
draft-cui-dhc-dhcpv6-yang-01

Abstract

There has no unified method to configure DHCPv6 server ,relay and client itself, always pre-configured manually by operators.

IETF netmod WG has developed a general data model for NETCONF protocol, YANG data model [RFC6020].

This document defines a YANG data model for the configuration and management of DHCPv6 server, DHCPv6 relay and DHCPv6 client. With this model, the operators can configure and manage the devices by using NETCONF.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

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 September 25, 2015.

Table of Contents

1. Introduction

This document defines a YANG data model for the configuration and management of DHCPv6 server, DHCPv6 relay and DHCPv6 client. With this model, the operators can configure and manage the devices by using NETCONF.

Model include three sub-modules:

- DHCPv6 server

- DHCPv6 relay

- DHCPv6 client

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The reader should be familiar with the terms defined in DHCPv6 [RFC3315] and relevant documents.

DHCPv6 tree diagrams provide a concise representation of a YANG module to help readers understand the module structure. The meaning if the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Parentheses "(" and ")" enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Symbols after data node names: "?" means an optional node, and "*" denotes a list and leaf-list.
- Abbreviations before data node names: "rw" means configuration data (read-write), and "ro" means state data (read-only).

2. Objectives

This document defines a YANG data model that can be used to configure and manage DHCPv6 server, DHCPv6 relay and DHCPv6 client.

2.1. DHCPv6 server

DHCPv6 server parameters.

2.2. DHCPv6 relay

DHCPv6 relay parameters.

2.3. DHCPv6 client

DHCPv6 client parameters.

3. DHCPv6 Tree Diagrams

3.1. DHCPv6 Server Tree Diagrams

```
+--rw dhcpv6
   +--rw server                                {dhcpv6-server}?
      +--rw servAttributes
      |  +--rw name                            string
      |  +--ro duid                            uint32
      |  +--rw enable                          boolean
      |  +--rw description?                    string
      |  +--rw pd-function                     boolean
      |  +--rw two-step-interaction            boolean
      |  +--rw rapidCommit                     boolean
      |  +--ro vendorInfo
      |     +--ro entNum                       uint8
      |     +--ro data*                        string
      +--rw address-pools
      |  +--rw address-pool* [pool-name]
      |  |  +--rw pool-name                    string
      |  |  +--rw pool-prefix                  inet:ipv6-prefix
      |  |  +--rw start-address                inet:ipv6-address-no-zone
      |  |  +--rw end-address                  inet:ipv6-address-no-zone
      |  |  +--rw preferred-lifetime           yang:timeticks
      |  |  +--rw valid-lifetime               yang:timeticks
      |  |  +--ro used-ipv6-count              uint32
      |  |  +--ro idle-ipv6-count              uint32
      |  +--ro bindingInfo* [cliDUID]
      |     +--ro cliDUID                      uint32
      |     +--ro cliIA* [iaid]
      |        +--ro iaType                    string
      |        +--ro iaid                      uint8
      |        +--ro cliAddr*                  inet:ipv6-address
      |        +--ro pool-name?                string
      +--rw prefix-pools
      |  +--rw prefix-pool* [pool-name]
      |  |  +--rw pool-name                    string
      |  |  +--rw prefix                       inet:ipv6-prefix
      |  |  +--rw prefix-length                uint8
      |  |  +--rw preferred-lifetime           yang:timeticks
      |  |  +--rw valid-lifetime               yang:timeticks
      |  +--ro bindingInfo* [cliDUID]
      |     +--ro cliDUID                      uint32
      |     +--ro cliIAPD* [iaid]
      |        +--ro iaid                      uint8
      |        +--ro cliPrefix*                inet:ipv6-prefix
      |        +--ro cliPrefixLen*             uint8
      |        +--ro pool-name?                string
      +--rw otherParas?
      |  +--rw dnsServer* [dnsName]
```

```
|  |  +--rw dnsName                      string
|  |  +--rw dnsServAddr                  inet:ipv6-address
|  +--rw domainSearchList                string
|  +--rw sipServer* [sipServDomainName]
|  |  +--rw sipServDomainName            string
|  |  +--rw sipServAddr                  inet:ipv6-address
|  +--rw sntpServer* [sntpServName]
|  |  +--rw sntpServName                 string
|  |  +--rw sntpServAddr                 inet:ipv6-address
|  +--rw ntpServParas* [ntpName]
|  |  +--rw ntpName                      string
|  |  +--rw ntpServAddr                  inet:ipv6-address
|  |  +--rw ntpServMulAddr               inet:ipv6-address
|  |  +--rw ntpServFQDN                  string
|  +--rw nisParas
|  |  +--rw nisServAddr                  inet:ipv6-address
|  |  +--rw nisCli* [cliDUID]
|  |     +--rw cliDUID                   uint32
|  |     +--rw cliDomainName             string
|  +--rw nispParas
|     +--rw nispServAddr                 inet:ipv6-address
|     +--rw nispCli* [cliDUID]
|        +--rw cliDUID                   uint32
|        +--rw cliDomainName             string
+--rw relayOpaqueParas?
|  +--rw relays* [relayName]
|     +--rw relayName                    string
|     +--rw interfaceInfo* [ifName]
|     |  +--rw ifName                    string
|     |  +--rw interface-id              string
|     +--rw subscribers* [subscriberName]
|     |  +--rw subscriberName            string
|     |  +--rw subscriber-id             string
|     +--rw remoteHost* [entNum]
|        +--rw entNum                    uint8
|        +--rw remote-id                 string
+--ro packetStats
   +--ro solicit-count                   uint32
   +--ro request-count                   uint32
   +--ro renew-count                     uint32
   +--ro rebind-count                    uint32
   +--ro decline-count                   uint32
   +--ro release-count                   uint32
   +--ro infoReq-count                   uint32
   +--ro advertise-count                 uint32
   +--ro confirm-count                   uint32
   +--ro reply-count                     uint32
   +--ro reconfigure-count               uint32
```

```
      +--ro relay-forward-count            uint32
      +--ro relay-reply-count              uint32
```

Figure 1: DHCPv6 Data Model Structure

Introduction of important nodes:

- servAttributes: This container contains basic attributes of a DHCPv6 server such as DUID, server name and so on. Some optional functions that can be provided by the server is also included.

- duid: Each server and client has only one DUID (DHCP Unique Identifier). The DUID here identifies a unique DHCPv6 server for clients.

- pd-function: Whether the server can act as a delegating router to perform prefix delegation ([RFC3633]).

- two-step-interaction : A boolean value specifies whether the server support client-server exchanges involving two messages defined in ([RFC3315]).

- rapidCommit: Setting the value to '1' represents the server support the Solicit-Reply message exchange. '0' means the server will simply ignore the Rapid Commit option in Solicit message.

- address-pools: A container describes the DHCPv6 server's address pools.

- address-pool: A DHCPv6 server can be configured with several address pools. This list defines such address pools which are distinguish by the key called "pool-name".

- bindingInfo: A list records a binding information for each DHCPv6 client that has already been allocated IPv6 addresses.

- prefix-pools: If a server supports prefix delegation function, this container will be used to define the delegating router's prefix pools.

- prefix-pool: Similar to server's address pools, a delegating router can also be configured with multiple prefix pools specified by a list called "prefix-pool".

- bindingInfo: A list records a binding information for each DHCPv6 requesting router that has already been configured IPv6 prefixes.

- otherParas: This container defines extra configuration parameters

provided by the DHCPv6 server apart from the address and prefix information. Such parameters may include DNS servers, SIP servers, SNTP servers, etc.

- relayOpaqueParas: This container contains some opaque values in Relay Agent options that need to be configured on the server side only for value match. Such Relay Agent options include Interface-Id option, Remote-Id option and Subscriber-Id option.

- packetStats: A container presents the packet statistics related to the DHCPv6 server.

3.2. DHCPv6 Relay Tree Diagrams

```
+--rw dhcpv6
   +-- ...
   |
   +--rw relay                                  {dhcpv6-relay}?
      +--rw relayAttributes
      |  +--rw name                             string
      |  +--rw enable                           boolean
      |  +--rw destAddrs*                       inet:ipv6-address
      |  +--rw subscribers* [subscriberName]?
      |  |  +--rw subscriberName                string
      |  |  +--rw subscriber-id                 string
      |  +--rw remoteHost* [entNum]?
      |  |  +--rw entNum                        uint8
      |  |  +--rw remote-id                     string
      |  +--ro vendorInfo
      |  |  +--ro entNum                        uint8
      |  |  +--ro data*                         string
      |  +--rw relayInterfaces
      |     +--rw relayIf* [ifName]
      |        +--rw ifName                     string
      |        +--rw enable                     boolean
      |        +--rw interface-id?              string
      |        +--rw nextEntity* [destAddr]
      |           +--rw destAddr                inet:ipv6-address
      |           +--rw available               boolean
      |           +--rw multicast               boolean
      |           +--rw server                  boolean
      |           +--ro packetStats
      |              +--ro cliPacketRvd-count   uint32
      |              +--ro solicitRvd-count     uint32
      |              +--ro requestRvd-count     uint32
      |              +--ro renewRvd-count       uint32
      |              +--ro rebindRvd-count      uint32
      |              +--ro declineRvd-count     uint32
```

```
        |              +--ro releaseRvd-count    uint32
        |              +--ro infoReqRvd-count    uint32
        |              +--ro relayForRvd-count   uint32
        |              +--ro relayRepRvd-count   uint32
        |              +--ro packetToCli-count   uint32
        |              +--ro advertiseSent-count uint32
        |              +--ro confirmSent-count   uint32
        |              +--ro replySent-count     uint32
        |              +--ro reconfigSent-count  uint32
        |              +--ro relayForSent-count  uint32
        |              +--ro relayRepSent-count  uint32
        +--ro relayStats
           +--ro cliPacketRvd-count              uint32
           +--ro relayForRvd-count               uint32
           +--ro relayRepRvd-count               uint32
           +--ro packetToCli-count               uint32
           +--ro relayForSent-count              uint32
           +--ro relayRepSent-count              uint32
           +--ro discardedPacket-count           uint32
```

Introduction of important nodes:

- relayAttributes: A container describes some basic attributes of the relay agent including some relay agent specific options data that need to be configured previously. Such options include Remote-Id option and Subscriber-Id option.

- destAddrs: Each DHCPv6 relay agent may be configured with a list of destination addresses. This node defines such a list of IPv6 addresses that may include unicast addresses, multicast addresses or other addresses.

- relayInterfaces: It is a sub-container of "relayAttributes" that defines common configuration and state parameters in the interfaces of a DHCPv6 relay agent.

- relayIf: A list describes a specific interface and its corresponding parameters. Here we use a string called "ifName" as the key of list.

- nextEntity: This node defines a list that is used to describe the next hop entity of this relay agent. Different entities are distinguished by their addresses.

- packetStats: A container shows packet state information of a specific data communication.

- relayStats: The "relayStats" container records and presents the

overall packet statistics of the relay agent.

3.3. DHCPv6 Client Tree Diagrams

```
   +--rw dhcpv6
      +-- ...
      |
      +--rw client                                  {dhcpv6-client}?
         +--rw clientInterfaces
            +--rw clientIf* [ifName]
               +--rw ifName                         string
               +--ro duid                           uint32
               +--rw enable                         boolean
               +--rw cliFQDN?                       string
               +--rw pd-function                    boolean
               +--rw rapidCommit                    boolean
               +--rw dual-stack                     boolean
               +--rw mo-tab
               |  +--rw m-tab                       boolean
               |  +--rw o-tab                       boolean
               +--ro vendorInfo
               |  +--ro entNum                      uint8
               |  +--ro data*                       string
               +--ro identity-associations
               |  +--ro identity-association* [iaid]
               |     +--ro iaid                     uint8
               |     +--ro iaType                   string
               |     +--ro ipv6Addr*                inet:ipv6-address
               |     +--ro ipv6Prefix*              inet:ipv6-prefix
               |     +--ro prefix-length*           uint8
               |     +--ro t1-time                  yang:date-and-time
               |     +--ro t2-time                  yang:date-and-time
               |     +--ro preferred-lifetime       yang:timeticks
               |     +--ro valid-lifetime           yang:timeticks
               +--ro ifOtherParas?
               |  +--ro dnsServAddr*                inet:ipv6-address
               |  +--ro domainSearchList            string
               |  +--ro sipServAddr*                inet:ipv6-address
               |  +--ro sipServDomainNameList       string
               |  +--ro uniDHCPv6ServAddr           inet:ipv6-address
               |  +--ro sntpServAddr*               inet:ipv6-address
               |  +--ro ntpServParas
               |  |  +--ro ntpServAddr              inet:ipv6-address
               |  |  +--ro ntpServMulAddr           inet:ipv6-address
               |  |  +--ro ntpServFQDN              string
               |  +--ro nisParas
               |  |  +--ro nisServAddr              inet:ipv6-address
               |  |  +--ro nisCliDomainName         string
```

```
|  +--ro nispParas
|     +--ro nispServAddr          inet:ipv6-address
|     +--ro nispCliDomainName     string
+--ro packetStats
   +--ro solicit-count            uint32
   +--ro request-count            uint32
   +--ro renew-count              uint32
   +--ro rebind-count             uint32
   +--ro decline-count            uint32
   +--ro release-count            uint32
   +--ro infoReq-count            uint32
   +--ro advertise-count          uint32
   +--ro confirm-count            uint32
   +--ro reply-count              uint32
   +--ro reconfigure-count        uint32
```

Introduction of important nodes:

- clientInterfaces: A client may have several interfaces, it is more reasonable to configure and manage parameters on the interface-level. This container includes configuration and state data of a DHCPv6 client in a per-interface manner.

- clientIf: The list defines a specific client interface and its data. Different interfaces are distinguished by the "ifName" key which is a configurable string value.

- duid: Each server and client has only one DUID (DHCP Unique Identifier). The DUID here will be carried in the Client ID option to identify a specific DHCPv6 client.

- cliFQDN: A DHCPv6 server needs to know the Fully Qualified Domain Name (FQDN) of the client to achieve the DNS update.

- pd-function: Whether the client can act as a requesting router to request prefixes using prefix delegation ([RFC3633]).

- rapidCommit: '1' indicates a client can initiate a Solicit-Reply message exchange by adding a Rapid Commit option in Solicit message. '0' means the client is not allowed to add a Rapid Commit option to request addresses in a two-message exchange pattern.

- mo-tab: The management tab label indicates the operation mode of the DHCPv6 client. 'm'=1 and 'o'=1 indicate the client will use DHCPv6 to obtain all the configuration data. 'm'=1 and 'o'=0 are a meaningless combination. 'm'=0 and 'o'=1 indicate the client will use stateless DHCPv6 to obtain configuration data apart from addresses/prefixes data. 'm'=0 and 'o'=0 represent the client will

not use DHCPv6 but use SLAAC to achieve configuration.

- identity-association: IA is a construct through which a server and a client can identify, group, and manage a set of related IPv6 addresses. The key of the "identity-association" list is a 4-byte number IAID defined in [RFC3315] .

- ifOtherParas: A client can obtain extra configuration data other than address and prefix information through DHCPv6. This container describes such data the client was configured. The potential configuration data may include DNS server addresses, SIP server domain names, etc.

- packetStats: A container records all the packet status information of a specific interface.

3.4. Notifications Mechanism for DHCPv6

```
   +--rw dhcpv6
      +-- ...
      |
      +--n notifications
         +--n dhcpv6-server-event                {dhcpv6-server}?
         |  +--n addr-used-up
         |  |  +--ro servDUID                    uint32
         |  |  +--ro servName?                   string
         |  |  +--ro pool-name                   string
         |  +--n prefix-used-up
         |  |  +--ro servDUID                    uint32
         |  |  +--ro servName?                   string
         |  |  +--ro pool-name                   string
         |  +--n invalid-client-detected
         |     +--ro cliDUID                     uint32
         |     +--ro description?                string
         +--n dhcpv6-relay-event                 {dhcpv6-relay}?
         |  +--n topo-changed
         |     +--ro relayIfName                 string
         |     +--ro first-hop                   boolean
         |     +--ro lastEntityAddr              inet:ipv6-address
         +--n dhcpv6-client-event                {dhcpv6-client}?
            +--n invalid-ia-detected
            |  +--ro cliDUID                     uint32
            |  +--ro iaid                        uint8
            |  +--ro servDUID?                   uint32
            |  +--ro description?                string
            +--n retransmission-failed
            |  +--ro cliDUID                     uint32
            |  +--ro description                 enum
            +--n failed-status-turn-up
               +--ro cliDUID                     uint32
               +--ro statusCode                  enum
```

Introduction of notifications:

- addr-used-up: raised when the address pool has run out all its addresses.

- prefix-used-up: raised when the prefix pool has run out all it prefixes.

- invalid-client-detected: raised when the server has found a client which can be regarded as a potential attacker. Some description could also be included.

- topo-changed: raised when the topology of the relay agent is changed.

- invalid-ia-detected: raised when the identity association of the client can be proved to be invalid. Possible condition includes duplicated address, illegal address, etc.

- retransmission-failed: raised when the retransmission mechanism defined in [RFC3315] is failed.

- failed-status-turn-up: raised when the client receives a message includes an unsuccessful Status Code option.

4. DHCPv6 YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6@2015-03-24.yang"

module ietf-dhcpv6 {
    namespace "urn:ietf:params:xml:ns:yang:dhcpv6";
    prefix "dhcpv6";

    import ietf-inet-types { prefix inet; revision-date "2013-07-15"; }
    import ietf-yang-types { prefix yang; revision-date "2013-07-15"; }

    organization "dhc wg";
    contact "yong@csnet1.cs.tsinghua.edu.cn
             wangh13@mails.tsinghua.edu.cn
             lh.sunlinh@gmail.com
             ";

    description "This model defines a YANG data model that can be used to
configure
        and manage DHCPv6 server, DHCPv6 relay and DHCPv6 client.";

    revision 2015-03-24 {
        description "Version01: Correct grammar errors, Reuse groupings, Update
            "dhcpv6-realy" feature, Add notifications.;
    }

    revision 2015-03-04 {
        description "Version00: Initial revision.";
    }

/*
 * Features
 */

    feature dhcpv6-server {
      description
        "Server in DHCPv6.";
```

```
      reference
        "RFC3315";
    }

    feature dhcpv6-relay {
      description
        "Relay agent in DHCPv6.";
      reference
        "RFC3315";
    }

    feature dhcpv6-client {
      description
        "Client in DHCPv6.";
      reference
        "RFC3315";
    }

/*
 * Grouping
 */

    grouping vendor-info {
        container vendorinfo {
            config "false";
            leaf entnum {
                type uint8;
            }
            leaf-list data {
                type string;
            }
        }
    }


/*
 * Data Nodes
 */

    container server {
        if-feature dhcpv6-server;
        container servattributes {
            description "This container contains basic attributes of a DHCPv6
server
                such as DUID, server name and so on. Some optional functions
that
                can be provided by the server is also included.";
            leaf name {
```

```
            type string;
        }
```

```
            leaf duid {
                config false;
                description "Each server and client has only one DUID (DHCP
Unique Identifier).
                    The DUID here identifies a unique DHCPv6 server for
clients.";
                type uint32;
            }
            leaf enable {
                type boolean;
            }
            leaf description {
                type string;
            }
            leaf pd-function {
                description "Whether the server can act as a delegating router
to perform
                    prefix delegation ([RFC3633]).";
                type boolean;
            }
            leaf two-step-interaction {
                description "A boolean value specifies whether the server
support client-server
                    exchanges involving two messages defined in ([RFC3315]).";
                type boolean;
            }
            leaf rapidcommit {
                description "A boolean value specifies whether the server
support client-server
                    exchanges involving two messages defined in ([RFC3315]).";
                type boolean;
            }
            uses vendor-info;
        }
        container address-pools {
            description "A container describes the DHCPv6 server's address
pools.";
            list address-pool {
                description "A DHCPv6 server can be configured with several
address pools.
                    This list defines such address pools which are distinguish
by
                    the key called 'pool-name'.";
                key pool-name;
                leaf pool-name {
                    type string;
                }
                leaf pool-prefix {
```

```
                type inet:ipv6-prefix;
            }
            leaf start-address {
                type inet:ipv6-address-no-zone;
            }
            leaf end-address {
                type inet:ipv6-address-no-zone;
            }
```

```
                leaf preferred-lifetime {
                    type yang:timeticks;
                }
                leaf valid-lifetime {
                    type yang:timeticks;
                }
                leaf used-ipv6-count {
                    config "false";
                    type uint32;
                }
                leaf idle-ipv6-count {
                    config "false";
                    type uint32;
                }
            }
            list bindinginfo {
                config "false";
                description "A list records a binding information for each
DHCPv6 client that
                    has already been allocated IPv6 addresses.";
                key cliduid;
                leaf cliduid {
                    type uint32;
                }
                list cliia {
                    key iaid;
                    leaf iaid {
                        type uint8;
                    }
                    leaf iatype {
                        type string;
                    }
                    leaf-list cliaddr {
                        type inet:ipv6-address;
                    }
                    leaf pool-name {
                        type string;
                    }
                }
            }
        }
        container prefix-pools {
            description "If a server supports prefix delegation function, this
container
                will be used to define the delegating router's refix pools.";
            list prefix-pool {
                description "Similar to server's address pools, a delegating
router can also
```

```
                    be configured with multiple prefix pools specified by a
list called
                    'prefix-pool'.";
                key pool-name;
```

```
                leaf pool-name {
                    description "";
                    type string;
                }
                leaf  prefix {
                    description "";
                    type inet:ipv6-prefix;
                }
                leaf prefix-length {
                    description "";
                    type uint8;
                }
                leaf preferred-lifetime {
                    description "";
                    type yang:timeticks;
                }
                leaf valid-lifetime {
                    description "";
                    type yang:timeticks;
                }
            }
            list bindinginfo {
                config "false";
                description "A list records a binding information for each
DHCPv6 client that
                    has already been allocated IPv6 addresses.";
                key cliduid;
                leaf cliduid {
                    type uint32;
                }
                list cliiapd {
                    key iaid;
                    leaf iaid {
                        type uint8;
                    }
                    leaf-list cliprefix {
                        type inet:ipv6-prefix;
                    }
                    leaf-list cliprefixlen {
                        type uint8;
                    }
                    leaf pool-name {
                        type string;
                    }
                }
            }
        }
        container otherparas {
```

description "This container defines extra configuration parameters provided

```
                by the DHCPv6 server apart from the address and prefix
information.
                Such parameters may include DNS servers, SIP servers, SNTP
servers,
                etc.";
            list dnsserver {
                key dnsname;
                leaf dnsname {
                    type string;
                }
                leaf dnsservaddr {
                    type inet:ipv6-address;
                }
            }
            leaf domainsearchlist {
                type string;
            }
            list sipserver {
                key sipservdomainname;
                leaf sipserverdomainname {
                    type string;
                }
                leaf sipservaddr {
                    type inet:ipv6-address;
                }
            }
            list sntpserver {
                key sntpservname;
                leaf sntpservname {
                    type string;
                }
                leaf sntpservaddr {
                    type inet:ipv6-address;
                }
            }
            list ntpservparas {
                key ntpname;
                leaf ntpname {
                    type string;
                }
                leaf ntpservaddr {
                    type inet:ipv6-address;
                }
                leaf ntpservmuladdr {
                    type inet:ipv6-address;
                }
                leaf ntpservfqdn {
                    type string;
```

```
            }
        }
```

```
            container nisparas {
                leaf nisservaddr {
                    type inet:ipv6-address;
                }
                list niscli {
                    key cliduid;
                    leaf cliduid {
                        type uint32;
                    }
                    leaf clidomainname {
                        type string;
                    }
                }
            }
            container nispparas {
                leaf nispservaddr {
                    type inet:ipv6-address;
                }
                list nispcli {
                    key cliduid;
                    leaf cliduid {
                        type uint32;
                    }
                    leaf clidomainname {
                        type string;
                    }
                }
            }
        }
        container relayopaqueparas {
            description "This container contains some opaque values in Relay
Agent options
                that need to be configured on the server side only for value
match.
                Such Relay Agent options include Interface-Id option, Remote-Id
                option and Subscriber-Id option.";
            list relays {
                key relayname;
                leaf relayname {
                    type string;
                }
                list interfaceinfo {
                    key ifname;
                    leaf ifname {
                        type string;
                    }
                    leaf interface-id {
                        type string;
```

```
                }
            }
```

```
                list subscribers {
                    key subscribername;
                    leaf subscribername {
                        type string;
                    }
                    leaf subscribernameid {
                        type string;
                    }
                }
                list remotehost {
                    key entnum;
                    leaf entnum {
                        type uint8;
                    }
                    leaf remote-id {
                        type string;
                    }
                }
            }
        }
        container packetstats {
            config "false";
            description "A container presents the packet statistics related to
the DHCPv6
                server.";
            leaf solicit-count {
                type uint32;
            }
            leaf request-count {
                type uint32;
            }
            leaf renew-count {
                type uint32;
            }
            leaf rebind-count {
                type uint32;
            }
            leaf decline-count {
                type uint32;
            }
            leaf release-count {
                type uint32;
            }
            leaf inforeq-count {
                type uint32;
            }
            leaf advertise-count {
                type uint32;
```

```
            }
```

```
            leaf confirm-count {
                type uint32;
            }
            leaf reply-count {
                type uint32;
            }
            leaf reconfigure-count {
                type uint32;
            }
            leaf relay-forward-count {
                type uint32;
            }
            leaf relay-reply-count {
                type uint32;
            }
        }
    }

    container relay {
        if-feature dhcpv6-relay;
        container relayattributes {
            description "A container describes some basic attributes of the
relay agent
                including some relay agent specific options data that need to
be configured
                previously. Such options include Remote-Id option and
Subscriber-Id option.";
            leaf name {
                type string;
            }
            leaf enable {
                type boolean;
            }
            leaf-list destaddrs {
                description "Each DHCPv6 relay agent may be configured with a
list of destination
                    addresses. This node defines such a list of IPv6 addresses
that
                    may include unicast addresses, multicast addresses or other
addresses.";
                type inet:ipv6-address;
            }
            list subscribers {
                key subscribername;
                leaf subscribername {
                    type string;
                }
                leaf subscriber-id {
```

```
                type string;
            }
        }
        list remotehost {
            key entnum;
            leaf entnum {
```

```
                    type uint8;
                }
                leaf remote-id {
                    type string;
                }
            }
            uses vendor-info;
            container relayinterfaces {
                description "It is a container that defines common
configuration and state
                    parameters in the interfaces of a DHCPv6 relay agent. In
this
                    YANG data model for DHCPv6 relay agent, the parameters are
configured
                    in a per-interface manner.";
                list relayif {
                    description "A list describes a specific interface and its
corresponding parameters.
                        Here we use a string called 'ifName' as the key of
list.";
                    key ifname;
                    leaf ifname {
                        type string;
                    }
                    leaf enable {
                        type boolean;
                    }
                    leaf interface-id {
                        type string;
                    }
                    list nextentity {
                        description "This node defines a list that is used to
describe the next hop
                            entity of this relay distinguished by their
addresses.";
                        key destaddr;
                        leaf destaddr {
                            type inet:ipv6-address;
                        }
                        leaf available {
                            type boolean;
                        }
                        leaf multicast {
                            type boolean;
                        }
                        leaf server {
                            type boolean;
                        }
```

```
                        container packetstats {
                            config "false";
                            description "A container shows packet state
information of a specific interface.
                                It is a sub-container of the 'relayInterfaces'
container.";
                            leaf clipacketrvd-count {
                                type uint32;
                            }
```

```
leaf solicitrvd-count {
    type uint32;
}
leaf requestrvd-count {
    type uint32;
}
leaf renewrvd-count {
    type uint32;
}
leaf rebindrvd-count {
    type uint32;
}
leaf declinervd-count {
    type uint32;
}
leaf releaservd-count {
    type uint32;
}
leaf inforeqrvd-count {
    type uint32;
}
leaf relayforvd-count {
    type uint32;
}
leaf relayreprvd-count {
    type uint32;
}
leaf packettocli-count {
    type uint32;
}
leaf advertisesent-count {
    type uint32;
}
leaf confirmsent-count {
    type uint32;
}
leaf replysent-count {
    type uint32;
}
leaf reconfigsent-count {
    type uint32;
}
leaf relayforsent-count {
    type uint32;
}
leaf relayrepsent-count {
    type uint32;
}
```

```
                        }
                    }
                }
            }
        }
        container relaystats {
            config "false";
            description "The container records and presents the overall packet
statistics
                of the relay agent.";
            leaf clipacketrvd-count {
                type uint32;
            }
            leaf relayforrvd-count {
                type uint32;
            }
            leaf relayreprvd-count {
                type uint32;
            }
            leaf packettocli-count {
                type uint32;
            }
            leaf relayforsent-count {
                type uint32;
            }
            leaf relayrepsent-count {
                type uint32;
            }
            leaf discardedpacket-count {
                type uint32;
            }
        }
    }

    container client {
        if-feature dhcpv6-client;
        container clientinterfaces {
            description "A client may have several interfaces, it is more
reasonable to
                configure and manage parameters on the interface-level. This
container
                includes configuration and state data of a DHCPv6 client in a
                per-interface manner.";
            list clientif {
                description "The list defines a specific client interface and
its data. Different
                    interfaces are distinguished by the key which is a
configurable string
```

```
        value.";
      key ifname;
      leaf ifname {
        type string;
      }
```

```
                leaf duid {
                    config "false";
                    description "Each server and client has only one DUID (DHCP
Unique Identifier).
                        The DUID here will be carried in the Client ID option
to identify
                        a specific DHCPv6 client.";
                    type uint32;
                }
                leaf enable {
                    type boolean;
                }
                leaf clifqdn {
                    description "A DHCPv6 server needs to know the Fully
Qualified Domain Name
                        (FQDN) of the client to achieve the DNS update.";
                    type string;
                }
                leaf pd-function {
                    description "Whether the client can act as a requesting
router to request
                        prefixes using prefix delegation ([RFC3633]).";
                    type boolean;
                }
                leaf rapidcommit {
                    description "'1' indicates a client can initiate a Solicit-
Reply message exchange
                        by adding a Rapid Commit option in Solicit message. '0'
means
                        the client is not allowed to add a Rapid Commit option
to request
                        addresses in a two-message exchange pattern.";
                    type boolean;
                }
                leaf dual-stack {
                    type boolean;
                }
                container mo-tab {
                    description "The management tab label indicates the
operation mode of the
                        DHCPv6 client. 'm'=1 and 'o'=1 indicate the client will
use DHCPv6
                        to obtain all the configuration data. 'm'=1 and 'o'=0
are a meaningless
                        combination. 'm'=0 and 'o'=1 indicate the client will
use stateless
                        DHCPv6 to obtain configuration data apart from
addresses/prefixes
```

```
                        data. 'm'=0 and 'o'=0 represent the client will not use
DHCPv6
                        but use SLAAC to achieve configuration.";
                    leaf m-tab {
                        type boolean;
                    }
                    leaf o-tab {
                        type boolean;
                    }
                }
                uses vendor-info;
                container identity-associations {
                    config "false";
```

```
                    description "IA is a construct through which a server and a
client can identify,
                        group, and manage a set of related IPv6 addresses. The
key of
                        the list is a 4-byte number IAID defined in
[RFC3315] .";
                    list identity-association {
                        key iaid;
                        leaf iaid {
                            type uint8;
                        }
                        leaf iatype {
                            type string;
                        }
                        leaf-list ipv6addr {
                            type inet:ipv6-address;
                        }
                        leaf-list ipv6prefix {
                            type inet:ipv6-prefix;
                        }
                        leaf-list prefix-length {
                            type uint8;
                        }
                        leaf t1-time {
                            type yang:date-and-time;
                        }
                        leaf t2-time {
                            type yang:date-and-time;
                        }
                        leaf preferred-lifetime {
                            type yang:timeticks;
                        }
                        leaf valid-lifetime {
                            type yang:timeticks;
                        }
                    }
                }
                container ifotherparas {
                    config "false";
                    description "A client can obtain extra configuration data
other than address
                        and prefix information through DHCPv6. This container
describes
                        such data the client was configured. The potential
configuration
                        data may include DNS server addresses, SIP server
domain names, etc.";
                    leaf-list dnsservaddr {
```

```
                    type inet:ipv6-address;
                }
                leaf domainsearchlist {
                    type string;
                }
                leaf-list sipservaddr {
                    type inet:ipv6-address;
```

```
                    }
                    leaf sipservdomainnamelist {
                        type string;
                    }
                    leaf unidhcpv6servaddr {
                        type inet:ipv6-address;
                    }
                    leaf-list sntpservaddr {
                        type inet:ipv6-address;
                    }
                    container ntpservparas {
                        leaf ntpservaddr {
                            type inet:ipv6-address;
                        }
                        leaf ntpservmuladdr {
                            type inet:ipv6-address;
                        }
                        leaf ntpservfqdn {
                            type string;
                        }
                    }
                    container nisparas {
                        leaf nisservaddr {
                            type inet:ipv6-address;
                        }
                        leaf nisclidomainname {
                            type string;
                        }
                    }
                    container nispparas {
                        leaf nispservaddr {
                            type inet:ipv6-address;
                        }
                        leaf nispclidomainname {
                            type string;
                        }
                    }
                }
                container packetstats {
                    config "false";
                    description "A container records all the packet status
information of a specific
                        interface.";
                    leaf solicit-count {
                        type uint32;
                    }
                    leaf request-count {
                        type uint32;
```

```
                    }
```

```
                    leaf renew-count {
                        type uint32;
                    }
                    leaf rebind-count {
                        type uint32;
                    }
                    leaf decline-count {
                        type uint32;
                    }
                    leaf release-count {
                        type uint32;
                    }
                    leaf inforeq-count {
                        type uint32;
                    }
                    leaf advertise-count {
                        type uint32;
                    }
                    leaf confirm-count {
                        type uint32;
                    }
                    leaf reply-count {
                        type uint32;
                    }
                    leaf reconfigure-count {
                        type uint32;
                    }
                }
            }
        }
    }

/*
 * Notifications
 */

    notification notifications {
        container dhcpv6-server-event {
            if-feature dhcpv6-server;
            container addr-used-up {
                leaf servduid {
                    type uint32;
                }
                leaf servname {
                    type string;
                }
                leaf pool-name {
                    type string;
```

```
                }
            }
            container prefix-used-up {
                leaf servduid {
                    type uint32;
                }
                leaf servname {
                    type string;
                }
                leaf pool-name {
                    type string;
                }
            }
            container invalid-client-detected {
                leaf cliduid {
                    type uint32;
                }
                leaf description {
                    type string;
                }
            }
        }
        container dhcpv6-relay-event {
            if-feature dhcpv6-relay;
            container topo-changed {
                leaf relayifname {
                    type string;
                }
                leaf first-hop {
                    type boolean;
                }
                leaf lastentityaddr {
                    type inet:ipv6-address;
                }
            }
        }
        container dhcpv6-client-event {
            if-feature dhcpv6-client;
            container invalid-ia-detected {
                leaf cliduid {
                    type uint32;
                }
                leaf iaid {
                    type uint8;
                }
                leaf servduid {
                    type uint32;
                }
```

```
                leaf description {
                    type string;
                }
            }
            container retransmission-failed {
                leaf cliduid {
                    type uint32;
                }
                leaf description {
                    type enumeration {
                        enum "MRC failed";
                        enum "MRD failed";
                    }
                }
            }
            container failed-status-turn-up {
                leaf cliduid {
                    type uint32;
                }
                leaf statuscode {
                    type enumeration {
                        enum "1" {
                            description "UnspecFail";
                        }
                        enum "2" {
                            description "NoAddrAvail";
                        }
                        enum "3" {
                            description "NoBinding";
                        }
                        enum "4" {
                            description "NotOnLink";
                        }
                        enum "5" {
                            description "UseMulticast";
                        }
                    }
                }
            }
        }
    }
}
<CODE ENDS>
```

5. Security Considerations (TBD)

TBD

6. IANA Considerations (TBD)

TBD

7. Acknowledgements (TBD)

TBD

8. Normative References

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

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

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

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011.

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

Authors' Addresses

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6260-3059
Email: yong@csnet1.cs.tsinghua.edu.cn

Hao Wang
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: wangh13@mails.tsinghua.edu.cn

Linhui Sun
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: lh.sunlinh@gmail.com