

DHC Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: November 19, 2015

Y. Cui  
H. Wang  
L. Sun  
Tsinghua University  
T. Lemon  
Nominum  
I. Farrer  
Deutsche Telekom AG  
May 18, 2015

YANG Data Model for DHCPv6 Configuration  
draft-cui-dhc-dhcpv6-yang-02

## 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 November 19, 2015.

## Copyright Notice

Copyright (c) 2015 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.

## Table of Contents

1. Introduction . . . . .	2
1.1. Terminology . . . . .	3
2. Objectives . . . . .	3
2.1. DHCPv6 server . . . . .	3
2.2. DHCPv6 relay . . . . .	3
2.3. DHCPv6 client . . . . .	3
3. DHCPv6 Tree Diagrams . . . . .	4
3.1. DHCPv6 Server Tree Diagrams . . . . .	4
3.2. DHCPv6 Relay Tree Diagrams . . . . .	8
3.3. DHCPv6 Client Tree Diagrams . . . . .	9
3.4. Notifications Mechanism for DHCPv6 . . . . .	12
4. DHCPv6 YANG Model . . . . .	14
5. Security Considerations (TBD) . . . . .	35
6. IANA Considerations (TBD) . . . . .	35
7. Acknowledgements (TBD) . . . . .	35
8. Normative References . . . . .	35
Authors' Addresses . . . . .	35

## 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:

- o DHCPv6 server

- o DHCPv6 relay
- o 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 of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.
- o Braces "{" and "}" enclose feature content.
- o Parentheses "(" and ")" enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Symbols after data node names: "?" means an optional node, and "\*" denotes a list and leaf-list.
- o 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 serv-attributes
      +--rw name string
      +--ro duid
        +--ro duid-type uint16
        +--ro duid-high-1 uint32
        +--ro duid-high-2 uint32
        +--ro duid-low-1 uint32
        +--ro duid-low-2 uint32
      +--rw enable boolean
      +--rw description? string
      +--rw pd-function boolean
      +--rw stateless-service boolean
      +--rw rapid-commit boolean
      +--ro vendor-info
        +--ro ent-num uint32
        +--ro data* string
    +--rw address-pools
      +--rw address-pool* [pool-id]
        +--rw pool-id uint8
        +--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 binding-info* [cli-id]
        +--ro cli-id uint32
        +--ro duid
          +--ro duid-type uint16
          +--ro duid-high-1 uint32
          +--ro duid-high-2 uint32
          +--ro duid-low-1 uint32
          +--ro duid-low-2 uint32
        +--ro cli-ia* [iaid]
          +--ro ia-type string
          +--ro iaid uint32
          +--ro cli-addr* inet:ipv6-address
          +--ro pool-id? uint8
    +--rw prefix-pools
      +--rw prefix-pool* [pool-id]
        +--rw pool-id uint8

```

```

|   |--rw prefix                               inet:ipv6-prefix
|   |--rw prefix-length                       uint8
|   |--rw preferred-lifetime                  yang:timeticks
|   |--rw valid-lifetime                      yang:timeticks
|--ro binding-info* [cli-id]
|   |--ro cli-id                             uint32
|   |--ro duid
|       |--ro duid-type                       uint16
|       |--ro duid-high-1                    uint32
|       |--ro duid-high-2                    uint32
|       |--ro duid-low-1                     uint32
|       |--ro duid-low-2                     uint32
|--ro cli-iapd* [iaid]
|   |--ro iaid                               uint32
|   |--ro cli-prefix*                         inet:ipv6-prefix
|   |--ro cli-prefix-len*                     uint8
|   |--ro pool-id?                           uint8
|--rw other-paras
|   |--rw dns-server* [dns-serv-id]
|       |--rw dns-serv-id                     uint8
|       |--rw dns-serv-addr                   inet:ipv6-address
|--rw domain-search-list                     string
|--rw sip-server* [sip-serv-id]
|   |--rw sip-serv-id                         uint8
|   |--rw sip-serv-domain-name                string
|   |--rw sip-serv-addr                       inet:ipv6-address
|--rw sntp-server* [sntp-serv-id]
|   |--rw sntp-serv-id                       uint8
|   |--rw sntp-serv-addr                     inet:ipv6-address
|--rw ntp-serv-paras* [ntp-serv-id]
|   |--rw ntp-serv-id                         uint8
|   |--rw ntp-serv-addr                       inet:ipv6-address
|   |--rw ntp-serv-mul-addr                   inet:ipv6-address
|   |--rw ntp-serv-fqdn                       string
|--rw nis-paras
|   |--rw nis-serv-addr                       inet:ipv6-address
|   |--rw nis-cli-info* [cli-id]
|       |--rw cli-id                         uint32
|       |--rw duid
|           |--rw duid-type                   uint16
|           |--rw duid-high-1                 uint32
|           |--rw duid-high-2                 uint32
|           |--rw duid-low-1                  uint32
|           |--rw duid-low-2                  uint32
|       |--rw cli-domain-name                 string
|--rw nisp-paras
|   |--rw nisp-serv-addr                       inet:ipv6-address
|--rw nisp-cli-info* [cli-id]

```

```

|         +--rw cli-id
|         +--rw duid
|           +--rw duid-type          uint16
|           +--rw duid-high-1       uint32
|           +--rw duid-high-2       uint32
|           +--rw duid-low-1        uint32
|           +--rw duid-low-2        uint32
|         +--rw cli-domain-name     string
+--rw relay-opaque-paras
|   +--rw relays* [relay-name]
|     +--rw relay-name              string
|     +--rw interface-info* [if-name]
|       +--rw if-name               string
|       +--rw interface-id          string
|     +--rw subscribers* [subscriber]
|       +--rw subscriber            uint8
|       +--rw subscriber-id         string
|     +--rw remote-host* [ent-num]
|       +--rw ent-num               uint32
|       +--rw remote-id             string
+--ro packet-stats
  +--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 info-req-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:

- o serv-attributes: 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.
- o duid: Each server and client has only one DUID (DHCP Unique Identifier). The DUID here identifies a unique DHCPv6 server for clients. DUID consists of a two-octet type field and an arbitrary length (no more than 128 bits) content field. This duid container

includes a "duid-type" leaf to specify the type and following four leaves to define the variable length content.

- o `pd-function`: Whether the server can act as a delegating router to perform prefix delegation ([RFC3633]).
- o `two-step-interaction` : A boolean value specifies whether the server support client-server exchanges involving two messages defined in ([RFC3315]).
- o `rapid-commit`: 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.
- o `address-pools`: A container describes the DHCPv6 server's address pools.
- o `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".
- o `binding-info`: A list records a binding information for each DHCPv6 client that has already been allocated IPv6 addresses.
- o `prefix-pools`: If a server supports prefix delegation function, this container will be used to define the delegating router's prefix pools.
- o `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".
- o `binding-info`: A list records a binding information for each DHCPv6 requesting router that has already been configured IPv6 prefixes.
- o `other-paras`: 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.
- o `relay-opaque-paras`: 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.
- o `packet-stats`: A container presents the packet statistics related to the DHCPv6 server.

## 3.2. DHCPv6 Relay Tree Diagrams

```

+--rw dhcpv6
  +-- ...
  |
  +--rw relay {dhcpv6-relay}?
    +--rw relay-attributes
      +--rw name string
      +--rw enable boolean
      +--rw dest-addr* inet:ipv6-address
      +--rw subscribers* [subscriber]
        | +--rw subscriber uint8
        | +--rw subscriber-id string
      +--rw remote-host* [entNum]
        | +--rw ent-num uint32
        | +--rw remote-id string
      +--ro vendor-info
        | +--ro ent-num uint32
        | +--ro data* string
      +--rw relay-interfaces
        +--rw relay-if* [if-name]
          +--rw if-name string
          +--rw enable boolean
          +--rw interface-id? string
          +--rw next-entity* [dest-addr]
            +--rw dest-addr inet:ipv6-address
            +--rw available boolean
            +--rw multicast boolean
            +--rw server boolean
            +--ro packet-stats
              +--ro cli-packet-rvd-count uint32
              +--ro solicit-rvd-count uint32
              +--ro request-rvd-count uint32
              +--ro renew-rvd-count uint32
              +--ro rebind-rvd-count uint32
              +--ro decline-rvd-count uint32
              +--ro release-rvd-count uint32
              +--ro info-req-rvd-count uint32
              +--ro relay-for-rvd-count uint32
              +--ro relay-rep-rvd-count uint32
              +--ro packet-to-cli-count uint32
              +--ro adver-sent-count uint32
              +--ro confirm-sent-count uint32
              +--ro reply-sent-count uint32
              +--ro reconfig-sent-count uint32
              +--ro relay-for-sent-count uint32
              +--ro relay-rep-sent-count uint32
          +--ro relay-stats

```



```

+--ro cli-packet-rvd-count          uint32
+--ro relay-for-rvd-count           uint32
+--ro relay-rep-rvd-count           uint32
+--ro packet-to-cli-count           uint32
+--ro relay-for-sent-count           uint32
+--ro relay-rep-sent-count           uint32
+--ro discarded-packet-count         uint32

```

#### Introduction of important nodes:

- o relay-attributes: 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.
- o dest-addrs: 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.
- o relay-interfaces: It is a sub-container of "relayAttributes" that defines common configuration and state parameters in the interfaces of a DHCPv6 relay agent.
- o relay-if: A list describes a specific interface and its corresponding parameters. Here we use a string called "ifName" as the key of list.
- o next-entity: 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.
- o packet-stats: A container shows packet state information of a specific data communication.
- o relay-stats: 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 client-interfaces
      +--rw client-if* [if-name]
        +--rw if-name          string
        +--ro duid

```

```

|   +--ro duid-type                               uint16
|   +--ro duid-high-1                             uint32
|   +--ro duid-high-2                             uint32
|   +--ro duid-low-1                              uint32
|   +--ro duid-low-2                              uint32
+--rw enable                                     boolean
+--rw cli-fqdn?                                  string
+--rw pd-function                                boolean
+--rw rapid-commit                              boolean
+--rw dual-stack                                boolean
+--rw mo-tab
|   +--rw m-tab                                   boolean
|   +--rw o-tab                                   boolean
+--ro vendor-info
|   +--ro ent-num                                 uint32
|   +--ro data*                                  string
+--ro identity-associations
|   +--ro identity-association* [iaid]
|       +--ro iaid                               uint32
|       +--ro ia-type                             string
|       +--ro ipv6-addr*                          inet:ipv6-address
|       +--ro ipv6-prefix*                        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 if-other-paras
|   +--ro dns-serv-addr*                          inet:ipv6-address
|   +--ro domain-search-list                      string
|   +--ro sip-serv-addr*                          inet:ipv6-address
|   +--ro sip-serv-domain-name-list               string
|   +--ro uni-dhcpv6-serv-addr                    inet:ipv6-address
|   +--ro sntp-serv-addr*                         inet:ipv6-address
|   +--ro ntp-serv-paras
|       |   +--ro ntp-serv-addr                    inet:ipv6-address
|       |   +--ro ntp-serv-mul-addr                inet:ipv6-address
|       |   +--ro ntp-serv-fqdn                    string
|   +--ro nis-paras
|       |   +--ro nis-serv-addr                    inet:ipv6-address
|       |   +--ro nis-cli-domain-name              string
|   +--ro nisp-paras
|       +--ro nisp-serv-addr                      inet:ipv6-address
|       +--ro nisp-cli-domain-name                string
+--ro packet-stats
|   +--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 info-req-count        uint32
+--ro advertise-count       uint32
+--ro confirm-count         uint32
+--ro reply-count           uint32
+--ro reconfigure-count     uint32

```

#### Introduction of important nodes:

- o **client-interfaces:** 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.
- o **client-if:** The list defines a specific client interface and its data. Different interfaces are distinguished by the "ifName" key which is a configurable string value.
- o **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. This container are same as the "duid" container in "dhcpv6-server" feature.
- o **cli-fqdn:** A DHCPv6 server needs to know the Fully Qualified Domain Name (FQDN) of the client to achieve the DNS update.
- o **pd-function:** Whether the client can act as a requesting router to request prefixes using prefix delegation ([RFC3633]).
- o **rapid-commit:** '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.
- o **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.
- o **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] .

- o if-other-paras: 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.
- o packet-stats: 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 duid
          +--ro duid-type uint16
          +--ro duid-high-1 uint32
          +--ro duid-high-2 uint32
          +--ro duid-low-1 uint32
          +--ro duid-low-2 uint32
          +--ro serv-name? string
          +--ro pool-name string
      +--n prefix-used-up
        +--ro duid
          +--ro duid-type uint16
          +--ro duid-high-1 uint32
          +--ro duid-high-2 uint32
          +--ro duid-low-1 uint32
          +--ro duid-low-2 uint32
          +--ro serv-name? string
          +--ro pool-name string
      +--n invalid-client-detected
        +--ro duid
          +--ro duid-type uint16
          +--ro duid-high-1 uint32
          +--ro duid-high-2 uint32
          +--ro duid-low-1 uint32
          +--ro duid-low-2 uint32
          +--ro description? string
      +--n dhcpv6-relay-event {dhcpv6-relay}?
        +--n topo-changed
          +--ro relay-if-name string
          +--ro first-hop boolean
          +--ro last-entity-addr inet:ipv6-address
      +--n dhcpv6-client-event {dhcpv6-client}?

```

```

+--n invalid-ia-detected
|   +--ro duid
|   |   +--ro duid-type                uint16
|   |   +--ro duid-high-1             uint32
|   |   +--ro duid-high-2             uint32
|   |   +--ro duid-low-1              uint32
|   |   +--ro duid-low-2              uint32
|   +--ro ia-id                        uint32
|   +--ro serv-name?                  string
|   +--ro description?                 string
+--n retransmission-failed
|   +--ro duid
|   |   +--ro duid-type                uint16
|   |   +--ro duid-high-1             uint32
|   |   +--ro duid-high-2             uint32
|   |   +--ro duid-low-1              uint32
|   |   +--ro duid-low-2              uint32
|   +--ro description                  enumeration
+--n failed-status-turn-up
|   +--ro duid
|   |   +--ro duid-type                uint16
|   |   +--ro duid-high-1             uint32
|   |   +--ro duid-high-2             uint32
|   |   +--ro duid-low-1              uint32
|   |   +--ro duid-low-2              uint32
|   +--ro status-code                  enumeration

```

#### Introduction of notifications:

- o addr-used-up: raised when the address pool has run out all its addresses.
- o prefix-used-up: raised when the prefix pool has run out all its prefixes.
- o 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.
- o topo-changed: raised when the topology of the relay agent is changed.
- o 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.
- o retransmission-failed: raised when the retransmission mechanism defined in [RFC3315] is failed.

- o 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@2015-04-13.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
           Ted.Lemon@nominum.com
           ian.farrar@telekom.de";

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

  revision 2015-7-17 {
    description "Version02: Add duid grouping, correct errors.";
  }

  revision 2015-04-13 {
    description "Version02: Correct grammar errors.";
  }

  revision 2015-04-02 {
    description "Version01: Correct grammar errors, Reuse groupings, Upd
ate '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 vendor-info {
    config "false";
    leaf ent-num {
      mandatory true;
      type uint32;
    }
    leaf-list data {
      type string;
    }
  }
}

grouping duid {
  container duid {
    description "Each server and client has only one DUID (DHCP Unique Identifier).";
    config "false";
    leaf duid-type {
      mandatory true;
      type uint16;
    }
    leaf duid-high-1 {
      mandatory true;
    }
  }
}
```



```

        type uint32;
    }
    leaf duid-high-2 {
        mandatory true;
        type uint32;
    }
    leaf duid-low-1 {
        mandatory true;
        type uint32;
    }
    leaf duid-low-2 {
        mandatory true;
        type uint32;
    }
}

}

/*
 * Data Nodes
 */

container server {
    if-feature dhcpv6-server;
    container serv-attributes {
        description "This container contains basic attributes of a DHCPv
6 server
        such as DUID, server name and so on. Some optional functions
that
        can be provided by the server is also included.";
        leaf name {
            mandatory true;
            type string;
        }
        uses duid;
        leaf enable {
            mandatory true;
            type boolean;
        }
        leaf description {
            type string;
        }
        leaf pd-function {
            description "Whether the server can act as a delegating rout
er to perform
            prefix delegation ([RFC3633]).";
            mandatory true;
            type boolean;
        }
        leaf stateless-service {
            description "A boolean value specifies whether the server su
pport client-server

```



```
        exchanges involving two messages defined in ([RFC3315]).
";
        mandatory true;
        type boolean;
    }
    leaf rapid-commit {
        description "A boolean value specifies whether the server su
pport client-server
        exchanges involving two messages defined in ([RFC3315]).
";
        mandatory true;
        type boolean;
    }
    uses vendor-info;
}
container address-pools {
    description "A container describes the DHCPv6 server's address p
ools.";
    list address-pool {
        description "A DHCPv6 server can be configured with several
address pools.
        This list defines such address pools which are distingui
sh by
        the key called 'pool-name'.";
        key pool-id;
        leaf pool-id {
            mandatory true;
            type uint8;
        }
        leaf pool-prefix {
            mandatory true;
            type inet:ipv6-prefix;
        }
        leaf start-address {
            mandatory true;
            type inet:ipv6-address-no-zone;
        }
        leaf end-address {
            mandatory true;
            type inet:ipv6-address-no-zone;
        }
        leaf preferred-lifetime {
            mandatory true;
            type yang:timeticks;
        }
        leaf valid-lifetime {
            mandatory true;
            type yang:timeticks;
        }
        leaf used-ipv6-count {
            config "false";
            mandatory true;
            type uint32;
        }
    }
}
```



```

        leaf idle-ipv6-count {
            config "false";
            mandatory true;
            type uint32;
        }
    }
    list binding-info {
        config "false";
        description "A list records a binding information for each D
HCPv6 client that
            has already been allocated IPv6 addresses.";
        key cli-id;
        leaf cli-id {
            mandatory true;
            type uint32;
        }
        uses duid;
        list cli-ia {
            key ia-id;
            leaf ia-type {
                mandatory true;
                type string;
            }
            leaf ia-id {
                mandatory true;
                type uint32;
            }
            leaf-list cli-addr {
                type inet:ipv6-address;
            }
            leaf pool-id {
                type uint8;
            }
        }
    }
}
container prefix-pools {
    description "If a server supports prefix delegation function, th
is 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-id;
        leaf pool-id {
            mandatory true;
            type uint8;
        }
        leaf prefix {

```



```

        mandatory true;
        type inet:ipv6-prefix;
    }
    leaf prefix-length {
        mandatory true;
        type uint8;
    }
    leaf preferred-lifetime {
        mandatory true;
        type yang:timeticks;
    }
    leaf valid-lifetime {
        mandatory true;
        type yang:timeticks;
    }
}
list binding-info {
    config "false";
    description "A list records a binding information for each D
HCPv6 client that
        has already been allocated IPv6 addresses.";
    key cli-id;
    leaf cli-id {
        mandatory true;
        type uiny32;
    }
    uses duid;
    list cli-iapd {
        key iaaid;
        leaf iaaid {
            mandatory true;
            type uint32;
        }
        leaf-list cli-prefix {
            type inet:ipv6-prefix;
        }
        leaf-list cli-prefix-len {
            type uint8;
        }
        leaf pool-id {
            type uint8;
        }
    }
}
}
container other-paras {
    description "This container defines extra configuration paramete
rs provided
        by the DHCPv6 server apart from the address and prefix infor
mation.
        Such parameters may include DNS servers, SIP servers, SNTP s
ervers,
```





```
    etc.";
  list dns-server {
    key dns-serv-id;
    leaf dns-serv-id {
      mandatory true;
      type uint8;
    }
    leaf dns-serv-addr {
      mandatory true;
      type inet:ipv6-address;
    }
  }
  leaf domain-search-list {
    mandatory true;
    type string;
  }
  list sip-server {
    key sip-serv-id;
    leaf sip-serv-id {
      mandatory true;
      type uint8;
    }
    leaf sip-serv-domain-name {
      mandatory true;
      type string;
    }
    leaf sip-serv-addr {
      mandatory true;
      type inet:ipv6-address;
    }
  }
  list sntp-server {
    key sntp-serv-id;
    leaf sntp-serv-id {
      mandatory true;
      type uint8;
    }
    leaf sntp-serv-addr {
      mandatory true;
      type inet:ipv6-address;
    }
  }
  list ntp-serv-paras {
    key ntp-serv-id;
    leaf ntp-serv-id {
      mandatory true;
      type uint8;
    }
  }
```

```
    leaf ntp-serv-addr {
        mandatory true;
        type inet:ipv6-address;
    }
    leaf ntp-serv-mul-addr {
        mandatory true;
        type inet:ipv6-address;
    }
    leaf ntp-serv-fqdn {
        mandatory true;
        type string;
    }
}
container nis-paras {
    leaf nis-serv-addr {
        mandatory true;
        type inet:ipv6-address;
    }
    list nis-cli-info {
        key cli-id;
        leaf cli-id {
            mandatory true;
            type uint32;
        }
        uses duid;
        leaf cli-domain-name {
            mandatory true;
            type string;
        }
    }
}
container nisp-paras {
    leaf nisp-serv-addr {
        mandatory true;
        type inet:ipv6-address;
    }
    list nisp-cli-info {
        key cli-id;
        leaf cli-id {
            mandatory true;
            type uint32;
        }
        uses duid;
        leaf cli-domain-name {
            mandatory true;
            type string;
        }
    }
}
```

```
    }
  }
  container relay-opaque-paras {
    description "This container contains some opaque values in Relay
Agent options
match.
-Id
    that need to be configured on the server side only for value
    Such Relay Agent options include Interface-Id option, Remote
    option and Subscriber-Id option.";
  list relays {
    key relay-name;
    leaf relay-name {
      mandatory true;
      type string;
    }
    list interface-info {
      key if-name;
      leaf if-name {
        mandatory true;
        type string;
      }
      leaf interface-id {
        mandatory true;
        type string;
      }
    }
    list subscribers {
      key subscriber;
      leaf subscriber {
        mandatory true;
        type string;
      }
      leaf subscriber-id {
        mandatory true;
        type string;
      }
    }
    list remote-host {
      key ent-num;
      leaf ent-num {
        mandatory true;
        type uint32;
      }
      leaf remote-id {
        mandatory true;
        type string;
      }
    }
  }
}
```



```
    container packet-stats {
        config "false";
        description "A container presents the packet statistics related
to the DHCPv6
        server.";
        leaf solicit-count {
            mandatory true;
            type uint32;
        }
        leaf request-count {
            mandatory true;
            type uint32;
        }
        leaf renew-count {
            mandatory true;
            type uint32;
        }
        leaf rebind-count {
            mandatory true;
            type uint32;
        }
        leaf decline-count {
            mandatory true;
            type uint32;
        }
        leaf release-count {
            mandatory true;
            type uint32;
        }
        leaf info-req-count {
            mandatory true;
            type uint32;
        }
        leaf advertise-count {
            mandatory true;
            type uint32;
        }
        leaf confirm-count {
            mandatory true;
            type uint32;
        }
        leaf reply-count {
            mandatory true;
            type uint32;
        }
        leaf reconfigure-count {
            mandatory true;
            type uint32;
        }
    }
```

```
        leaf relay-forward-count {
            mandatory true;
            type uint32;
        }
        leaf relay-reply-count {
            mandatory true;
            type uint32;
        }
    }
}

container relay {
    if-feature dhcpv6-relay;
    container relay-attributes {
        description "A container describes some basic attributes of the
relay agent
        including some relay agent specific options data that need t
o be configured
        previously. Such options include Remote-Id option and Subscr
iber-Id option.";
        leaf name {
            mandatory true;
            type string;
        }
        leaf enable {
            mandatory true;
            type boolean;
        }
        leaf-list dest-addr {
            description "Each DHCPv6 relay agent may be configured with
a list of destination
            addresses. This node defines such a list of IPv6 address
es that
            may include unicast addresses, multicast addresses or ot
her addresses.";
            type inet:ipv6-address;
        }
        list subscribers {
            key subscriber;
            leaf subscriber {
                mandatory true;
                type string;
            }
            leaf subscriber-id {
                mandatory true;
                type string;
            }
        }
        list remote-host {
            key ent-num;
            leaf ent-num {
                mandatory true;
                type uint32;
            }
        }
    }
}
```



```

        leaf remote-id {
            mandatory true;
            type string;
        }
    }
    uses vendor-info;
    container relay-interfaces {
        description "It is a container that defines common configura
tion and state
        parameters in the interfaces of a DHCPv6 relay agent. In
        this
        YANG data model for DHCPv6 relay agent, the parameters a
re configured
        in a per-interface manner.";
        list relay-if {
            description "A list describes a specific interface and i
ts corresponding parameters.
            Here we use a string called 'ifName' as the key of l
ist.";
            key if-name;
            leaf if-name {
                mandatory true;
                type string;
            }
            leaf enable {
                mandatory true;
                type boolean;
            }
            leaf interface-id {
                type string;
            }
            list next-entity {
                description "This node defines a list that is used t
o describe the next hop
                entity of this relay distinguished by their addr
esses.";
                key dest-addr;
                leaf dest-addr {
                    mandatory true;
                    type inet:ipv6-address;
                }
                leaf available {
                    mandatory true;
                    type boolean;
                }
                leaf multicast {
                    mandatory true;
                    type boolean;
                }
                leaf server {
                    mandatory true;
                    type boolean;
                }
                container packet-stats {

```



```
config "false";
```

Cui, et al.

Expires November 19, 2015

[Page 25]

description "A container shows packet state information of a specific interface.

It is a sub-container of the 'relayInterface s' container.";

```
leaf cli-packet-rvd-count {
    mandatory true;
    type uint32;
}
leaf solicit-rvd-count {
    mandatory true;
    type uint32;
}
leaf request-rvd-count {
    mandatory true;
    type uint32;
}
leaf renew-rvd-count {
    mandatory true;
    type uint32;
}
leaf rebind-rvd-count {
    mandatory true;
    type uint32;
}
leaf decline-rvd-count {
    mandatory true;
    type uint32;
}
leaf release-rvd-count {
    mandatory true;
    type uint32;
}
leaf info-req-rvd-count {
    mandatory true;
    type uint32;
}
leaf relay-for-rvd-count {
    mandatory true;
    type uint32;
}
leaf relay-rep-rvd-count {
    mandatory true;
    type uint32;
}
leaf pac-to-cli-count {
    mandatory true;
    type uint32;
}
leaf adver-sent-count {
    mandatory true;
```



```

        type uint32;
    }
    leaf confirm-sent-count {
        mandatory true;
        type uint32;
    }
    leaf reply-sent-count {
        mandatory true;
        type uint32;
    }
    leaf reconfig-sent-count {
        mandatory true;
        type uint32;
    }
    leaf relay-for-sent-count {
        mandatory true;
        type uint32;
    }
    leaf relay-rep-sent-count {
        mandatory true;
        type uint32;
    }
}

}

}

}

container relay-stats {
    config "false";
    description "The container records and presents the overall pack
et statistics
of the relay agent.";
    leaf cli-packet-rvd-count {
        mandatory true;
        type uint32;
    }
    leaf relay-for-rvd-count {
        mandatory true;
        type uint32;
    }
    leaf relay-rep-rvd-count {
        mandatory true;
        type uint32;
    }
    leaf packet-to-cli-count {
        mandatory true;
        type uint32;
    }
    leaf relay-for-sent-count {

```

```

        mandatory true;
        type uint32;
    }
    leaf relay-rep-sent-count {
        mandatory true;
        type uint32;
    }
    leaf discarded-packet-count {
        mandatory true;
        type uint32;
    }
}

container client {
    if-feature dhcpv6-client;
    container client-interfaces {
        description "A client may have several interfaces, it is more re
asonable to
        container
            configure and manage parameters on the interface-level. This
            includes configuration and state data of a DHCPv6 client in
a
            per-interface manner.";
        list client-if {
            description "The list defines a specific client interface an
d its data. Different
            interfaces are distinguished by the key which is a confi
gurable string
                value.";
            key if-name;
            leaf if-name {
                mandatory true;
                type string;
            }
            uses duid;
            leaf enable {
                mandatory true;
                type boolean;
            }
            leaf cli-fqdn {
                description "A DHCPv6 server needs to know the Fully Qua
lified 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]).";
                mandatory true;
                type boolean;
            }
            leaf rapid-commit {

```

description "'1' indicates a client can initiate a Solic  
it-Reply message exchange

Cui, et al.

Expires November 19, 2015

[Page 28]

by adding a Rapid Commit option in Solicit message.

'0' means

the client is not allowed to add a Rapid Commit opti

on to request

```
addresses in a two-message exchange pattern.";
mandatory true;
type boolean;
```

```
}
leaf dual-stack {
  mandatory true;
  type boolean;
}
```

```
container mo-tab {
  description "The management tab label indicates the oper
```

ation mode of the

DHCPv6 client. 'm'=1 and 'o'=1 indicate the client w

ill 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 wil

l use stateless

DHCPv6 to obtain configuration data apart from addre

sses/prefixes

data. 'm'=0 and 'o'=0 represent the client will not

use DHCPv6

```
but use SLAAC to achieve configuration.";
leaf m-tab {
  mandatory true;
  type boolean;
```

```
}
leaf o-tab {
  mandatory true;
  type boolean;
```

```
}
}
uses vendor-info;
container identity-associations {
```

config "false";  
description "IA is a construct through which a server an  
d a client can identify,

group, and manage a set of related IPv6 addresses. T

he key of

the list is a 4-byte number IAID defined in [RFC3315

] .";

```
list identity-association {
  key iaid;
  leaf iaid {
    mandatory true;
    type uint32;
  }
  leaf ia-type {
    mandatory true;
    type string;
  }
}
```

```
leaf-list ipv6-addr {  
    type inet:ipv6-address;  
}  
leaf-list ipv6-prefix {  
    type inet:ipv6-prefix;
```



```

    }
    leaf-list prefix-length {
        type uint8;
    }
    leaf t1-time {
        mandatory true;
        type yang:date-and-time;
    }
    leaf t2-time {
        mandatory true;
        type yang:date-and-time;
    }
    leaf preferred-lifetime {
        mandatory true;
        type yang:timeticks;
    }
    leaf valid-lifetime {
        mandatory true;
        type yang:timeticks;
    }
}
}
container if-other-paras {
    config "false";
    description "A client can obtain extra configuration dat
a other than address
r describes
onfiguration
main names, etc.";
    leaf-list dns-serv-addr {
        type inet:ipv6-address;
    }
    leaf domain-search-list {
        mandatory true;
        type string;
    }
    leaf-list sip-serv-addr {
        type inet:ipv6-address;
    }
    leaf sip-serv-domain-name-list {
        mandatory true;
        type string;
    }
    leaf uni-dhcpv6-serv-addr {
        mandatory true;
        type inet:ipv6-address;
    }
    leaf-list sntp-serv-addr {
        type inet:ipv6-address;
    }

```

and prefix information through DHCPv6. This containe  
such data the client was configured. The potential c  
data may include DNS server addresses, SIP server do



```

    }
    container ntp-serv-paras {
        leaf ntp-serv-addr {
            mandatory true;
            type inet:ipv6-address;
        }
        leaf ntp-serv-mul-addr {
            mandatory true;
            type inet:ipv6-address;
        }
        leaf ntp-serv-fqdn {
            mandatory true;
            type string;
        }
    }
    container nis-paras {
        leaf nis-serv-addr {
            mandatory true;
            type inet:ipv6-address;
        }
        leaf nis-cli-domain-name {
            mandatory true;
            type string;
        }
    }
    container nisp-paras {
        leaf nisp-serv-addr {
            mandatory true;
            type inet:ipv6-address;
        }
        leaf nisp-cli-domain-name {
            mandatory true;
            type string;
        }
    }
}
container packet-stats {
    config "false";
    description "A container records all the packet status i
nformation of a specific
        interface.";
    leaf solicit-count {
        mandatory true;
        type uint32;
    }
    leaf request-count {
        mandatory true;
        type uint32;
    }
}

```

```
    leaf renew-count {
        mandatory true;
        type uint32;
    }
    leaf rebind-count {
        mandatory true;
        type uint32;
    }
    leaf decline-count {
        mandatory true;
        type uint32;
    }
    leaf release-count {
        mandatory true;
        type uint32;
    }
    leaf info-req-count {
        mandatory true;
        type uint32;
    }
    leaf advertise-count {
        mandatory true;
        type uint32;
    }
    leaf confirm-count {
        mandatory true;
        type uint32;
    }
    leaf reply-count {
        mandatory true;
        type uint32;
    }
    leaf reconfigure-count {
        mandatory true;
        type uint32;
    }
}
```

```
}
```

```
}
```

```
}
```

```
}
```

```
/*
 * Notifications
 */
```

```
notification notifications {
    container dhcpv6-server-event {
        if-feature dhcpv6-server;
```

```
    container addr-used-up {
      uses duid;
      leaf serv-name {
        type string;
      }
      leaf pool-name {
        mandatory true;
        type string;
      }
    }
    container prefix-used-up {
      uses duid;
      leaf serv-name {
        type string;
      }
      leaf pool-name {
        mandatory true;
        type string;
      }
    }
    container invalid-client-detected {
      uses duid;
      leaf description {
        type string;
      }
    }
  }
  container dhcpv6-relay-event {
    if-feature dhcpv6-relay;
    container topo-changed {
      leaf relay-if-name {
        mandatory true;
        type string;
      }
      leaf first-hop {
        mandatory true;
        type boolean;
      }
      leaf last-entity-addr {
        mandatory true;
        type inet:ipv6-address;
      }
    }
  }
  container dhcpv6-client-event {
    if-feature dhcpv6-client;
    container invalid-ia-detected {
      uses duid;
    }
  }
```

```
leaf iaaid {
    mandatory true;
    type uint32;
}
leaf serv-name {
    type string;
}
leaf description {
    type string;
}
}
container retransmission-failed {
    uses duid;
    leaf description {
        mandatory true;
        type enumeration {
            enum "MRC failed";
            enum "MRD failed";
        }
    }
}
container failed-status-turn-up {
    uses duid;
    leaf status-code {
        mandatory true;
        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: wanghl3@mails.tsinghua.edu.cn

Linhui Sun  
Tsinghua University  
Beijing 100084  
P.R.China

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

Ted Lemon  
Nominum, Inc.  
950 Charter St.  
Redwood City, CA 94043  
USA

Email: Ted.Lemon@nominum.com

Ian Farrer  
Deutsche Telekom AG  
CTO-ATI, Landgrabenweg 151  
Bonn, NRW 53227  
Germany

Email: ian.farrer@telekom.de