DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 26, 2018

Y. Cui
L. Sun
Tsinghua University
I. Farrer
S. Zechlin
Deutsche Telekom AG
Z. He
Tsinghua University
December 23, 2017

YANG Data Model for DHCPv6 Configuration
draft-ietf-dhc-dhcpv6-yang-05

Abstract

This document describes a YANG data model [RFC6020] for the configuration and management of DHCPv6 servers, relays, and clients.

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 https://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 June 26, 2018.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Terminology 3
2. DHCPv6 Tree Diagram 3
2.1. DHCPv6 Server Tree Diagrams 3
2.2. DHCPv6 Relay Tree Diagrams 16
2.3. DHCPv6 Client Tree Diagrams 20
3. DHCPv6 YANG Model 27
3.1. DHCPv6 Server YANG Model 27
3.2. DHCPv6 Relay YANG Model 48
3.3. DHCPv6 Client YANG Model 58
3.4. DHCPv6 Options YANG Model 67
4. Security Considerations (TBD) 98
5. IANA Considerations (TBD) 98
6. Acknowledgements 98
7. Contributors 98
8. References 98
8.1. Normative References 98
8.2. Informative References 99
Authors' Addresses 101

1. Introduction

DHCPv6 [RFC3315] is widely used for supplying configuration and other relevant parameters to clients in IPv6 networks. This document defines a DHCPv6 YANG data model, containing sub-modules for the configuration and management of DHCPv6 servers, relays and clients. A single YANG model covering all of these elements provides an operator with a common interface for the management of the entire DHCPv6 deployment in their network.

Since the publication of the original DHCPv6 specification, there have been a large number of addional documents that update the protocol's operation, add new functions and define new options. The YANG model described in this document incorporates all relevent changes. A full list of the documents which have been considedered in the development of this model is included in Appendix A.

IF - Commment - Does anyone have this list?

It is worth noting that as DHCPv6 is itself a device configuration protocol, it is not the intention of this document to replace the configuration of DHCPv6 options and parameters using the DHCPv6 protocol with the configuration of DHCPv6 options using NETCONF/YANG. The DHCPv6 client model is intended for the configuration of the DHCPv6 client function and also for obtaining read-only state data from the client which has been learnt via the normal DHCPv6 message flow. This gives an operator a better method for managing DHCPv6 clients and simplifies troubleshooting.

1.1. Terminology

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

The DHCPv6 tree diagrams provide a concise representation of a YANG module to help the reader understand the module structure.

A simplified graphical representation of the data model is provided in this d cument. For a description of the symbols in these diagrams, please refer to [I-D.ietf-netmod-yang-tree-diagrams].

2. DHCPv6 Tree Diagram

2.1. DHCPv6 Server Tree Diagrams

```
module: ietf-dhcpv6-server
+--rw server!
 +--rw server-config
 |  +--rw serv-attributes
 |  |  +--rw name?                string
 |  |  +--rw duid
 |  |  |  +--rw type-code?                  uint16
 |  |  |  +--rw (duid-type)?
 |  |  |     +--:(duid-llt)
 |  |  |     |  +--rw duid-llt-hardware-type?      uint16
 |  |  |     |  +--rw duid-llt-time?               yang:timeticks
 |  |  |     |  +--rw duid-llt-link-layer-addr?    yang:mac-address
 |  |  |     +--:(duid-en)
 |  |  |     |  +--rw duid-en-enterprise-number?   uint32
 |  |  |     |  +--rw duid-en-identifier?          string
 |  |  |     +--:(duid-ll)
 |  |  |     |  +--rw duid-ll-hardware-type?       uint16
 |  |  |     |  +--rw duid-ll-link-layer-addr?     yang:mac-address
 |  |  |     +--:(duid-uuid)
 |  |  |     |  +--rw uuid?                        yang:uuid
```

```
|  |  |     +--:(duid-invalid)
|  |  |        +--rw data?                          binary
|  |  +--rw ipv6-address*        inet:ipv6-address
|  |  +--rw description?         string
|  |  +--rw pd-function          boolean
|  |  +--rw stateless-service    boolean
|  |  +--rw rapid-commit         boolean
|  |  +--rw interfaces-config*   string
|  |  +--rw vendor-info
|  |     +--rw ent-num    uint32
|  |     +--rw data*      string
|  +--rw option-sets
|  |  +--rw option-set* [id]
|  |   +--rw id                                    uint32
|  |   +--rw server-unicast-option! {server-unicast-op}?
|  |   |  +--rw server-address?   inet:ipv6-address
|  |   +--rw sip-server-domain-name-list-option!
                          {sip-server-domain-name-list-op}?
|  |   |  +--rw sip-serv-domain-name    string
|  |   +--rw sip-server-address-list-option! {sip-server-address-list-op}?
|  |   |  +--rw sip-server* [sip-serv-id]
|  |   |     +--rw sip-serv-id      uint8
|  |   |     +--rw sip-serv-addr    inet:ipv6-address
|  |   +--rw dns-config-option! {dns-config-op}?
|  |   |  +--rw dns-server* [dns-serv-id]
|  |   |     +--rw dns-serv-id      uint8
|  |   |     +--rw dns-serv-addr    inet:ipv6-address
|  |   +--rw domain-searchlist-option! {domain-searchlist-op}?
|  |   |  +--rw domain-searchlist* [domain-searchlist-id]
|  |   |     +--rw domain-searchlist-id        uint8
|  |   |     +--rw domain-search-list-entry    string
|  |   +--rw nis-config-option! {nis-config-op}?
|  |   |  +--rw nis-server* [nis-serv-id]
|  |   |     +--rw nis-serv-id      uint8
|  |   |     +--rw nis-serv-addr    inet:ipv6-address
|  |   +--rw nis-plus-config-option! {nis-plus-config-op}?
|  |   |  +--rw nis-plus-server* [nis-plus-serv-id]
|  |   |     +--rw nis-plus-serv-id      uint8
|  |   |     +--rw nis-plus-serv-addr    inet:ipv6-address
|  |   +--rw nis-domain-name-option! {nis-domain-name-op}?
|  |   |  +--rw nis-domain-name?   string
|  |   +--rw nis-plus-domain-name-option!{nis-plus-domain-name-op}?
|  |   |  +--rw nis-plus-domain-name?   string
|  |   +--rw sntp-server-option! {sntp-server-op}?
|  |   |  +--rw sntp-server* [sntp-serv-id]
|  |   |     +--rw sntp-serv-id      uint8
|  |   |     +--rw sntp-serv-addr    inet:ipv6-address
|  |   +--rw info-refresh-time-option! {info-refresh-time-op}?
```

```
|  |   |  +--rw info-refresh-time    yang:timeticks
|  |   +--rw client-fqdn-option! {client-fqdn-op}?
|  |   |  +--rw server-initiate-update    boolean
|  |   |  +--rw client-initiate-update    boolean
|  |   |  +--rw modify-name-from-cli      boolean
|  |   +--rw posix-timezone-option! {posix-timezone-op}?
|  |   |  +--rw tz-posix    string
|  |   +--rw tzdb-timezone-option! {tzdb-timezone-op}?
|  |   |  +--rw tz-database    string
|  |   +--rw ntp-server-option! {ntp-server-op}?
|  |   |  +--rw ntp-server* [ntp-serv-id]
|  |   |     +--rw ntp-serv-id                    uint8
|  |   |     +--rw (ntp-time-source-suboption)?
|  |   |        +--:(server-address)
|  |   |        |  +--rw ntp-serv-addr-suboption* inet:ipv6-address
|  |   |        +--:(server-multicast-address)
|  |   |        |  +--rw ntp-serv-mul-addr-suboption* inet:ipv6-address
|  |   |        +--:(server-fqdn)
|  |   |           +--rw ntp-serv-fqdn-suboption*       string
|  |   +--rw boot-file-url-option! {boot-file-url-op}?
|  |   |  +--rw boot-file* [boot-file-id]
|  |   |     +--rw boot-file-id          uint8
|  |   |     +--rw suitable-arch-type*   uint16
|  |   |     +--rw suitable-net-if*      uint32
|  |   |     +--rw boot-file-url         string
|  |   +--rw boot-file-param-option! {boot-file-param-op}?
|  |   |  +--rw boot-file-paras* [para-id]
|  |   |     +--rw para-id      uint8
|  |   |     +--rw parameter    string
|  |   +--rw aftr-name-option! {aftr-name-op}?
|  |   |  +--rw tunnel-endpoint-name    string
|  |   +--rw kbr-default-name-option! {kbr-default-name-op}?
|  |   |  +--rw default-realm-name    string
|  |   +--rw kbr-kdc-option! {kbr-kdc-op}?
|  |   |  +--rw kdc-info* [kdc-id]
|  |   |     +--rw kdc-id            uint8
|  |   |     +--rw priority          uint16
|  |   |     +--rw weight            uint16
|  |   |     +--rw transport-type    uint8
|  |   |     +--rw port-number       uint16
|  |   |     +--rw kdc-ipv6-addr     inet:ipv6-address
|  |   |     +--rw realm-name        string
|  |   +--rw sol-max-rt-option! {sol-max-rt-op}?
|  |   |  +--rw sol-max-rt-value    yang:timeticks
|  |   +--rw inf-max-rt-option! {inf-max-rt-op}?
|  |   |  +--rw inf-max-rt-value    yang:timeticks
|  |   +--rw addr-selection-option! {addr-selection-op}?
|  |   |  +--rw a-bit-set       boolean
```

```
|  |   |  +--rw p-bit-set       boolean
|  |   |  +--rw policy-table* [policy-id]
|  |   |     +--rw policy-id     uint8
|  |   |     +--rw label         uint8
|  |   |     +--rw precedence    uint8
|  |   |     +--rw prefix-len    uint8
|  |   |     +--rw prefix        inet:ipv6-prefix
|  |   +--rw pcp-server-option! {pcp-server-op}?
|  |   |  +--rw pcp-server* [pcp-serv-id]
|  |   |     +--rw pcp-serv-id      uint8
|  |   |     +--rw pcp-serv-addr    inet:ipv6-address
|  |   +--rw s46-rule-option! {s46-rule-op}?
|  |   |  +--rw s46-rule* [rule-id]
|  |   |     +--rw rule-id           uint8
|  |   |     +--rw rule-type         enumeration
|  |   |     +--rw prefix4-len       uint8
|  |   |     +--rw ipv4-prefix       inet:ipv4-prefix
|  |   |     +--rw prefix6-len       uint8
|  |   |     +--rw ipv6-prefix       inet:ipv6-prefix
|  |   |     +--rw port-parameter
|  |   |        +--rw offset      uint8
|  |   |        +--rw psid-len    uint8
|  |   |        +--rw psid        uint16
|  |   +--rw s46-br-option! {s46-br-op}?
|  |   |  +--rw br* [br-id]
|  |   |     +--rw br-id           uint8
|  |   |     +--rw br-ipv6-addr    inet:ipv6-address
|  |   +--rw s46-dmr-option! {s46-dmr-op}?
|  |   |  +--rw dmr* [dmr-id]
|  |   |     +--rw dmr-id             uint8
|  |   |     +--rw dmr-prefix-len     uint8
|  |   |     +--rw dmr-ipv6-prefix    inet:ipv6-prefix
|  |   +--rw s46-v4-v6-binding-option! {s46-v4-v6-binding-op}?
|  |   |  +--rw ce* [ce-id]
|  |   |     +--rw ce-id              uint8
|  |   |     +--rw ipv4-addr          inet:ipv4-address
|  |   |     +--rw bind-prefix6-len    uint8
|  |   |     +--rw bind-ipv6-prefix    inet:ipv6-prefix
|  |   |     +--rw port-parameter
|  |   |        +--rw offset      uint8
|  |   |        +--rw psid-len    uint8
|  |   |        +--rw psid        uint16
|  |   +--rw operator-option-ipv6-address!{operator-op-ipv6-address}?
|  |   |  +--rw operator-ipv6-addr* [operator-ipv6-addr-id]
|  |   |     +--rw operator-ipv6-addr-id    uint8
|  |   |     +--rw operator-ipv6-addr       inet:ipv6-address
|  |   +--rw operator-option-single-flag!{operator-op-single-flag}?
|  |   |  +--rw flag* [flag-id]
```

```
|  |   |     +--rw flag-id       uint8
|  |   |     +--rw flag-value    boolean
|  |   +--rw operator-option-ipv6-prefix!{operator-op-ipv6-prefix}?
|  |   |  +--rw operator-ipv6-prefix* [operator-ipv6-prefix-id]
|  |   |     +--rw operator-ipv6-prefix-id      uint8
|  |   |     +--rw operator-ipv6-prefix6-len    uint8
|  |   |     +--rw operator-ipv6-prefix         inet:ipv6-prefix
|  |   +--rw operator-option-int32! {operator-op-int32}?
|  |   |  +--rw int32val* [int32val-id]
|  |   |     +--rw int32val-id    uint8
|  |   |     +--rw int32val       uint32
|  |   +--rw operator-option-int16! {operator-op-int16}?
|  |   |  +--rw int16val* [int16val-id]
|  |   |     +--rw int16val-id    uint8
|  |   |     +--rw int16val       uint16
|  |   +--rw operator-option-int8! {operator-op-int8}?
|  |   |  +--rw int8val* [int8val-id]
|  |   |     +--rw int8val-id    uint8
|  |   |     +--rw int8val       uint8
|  |   +--rw operator-option-uri! {operator-op-uri}?
|  |   |  +--rw uri* [uri-id]
|  |   |     +--rw uri-id    uint8
|  |   |     +--rw uri       string
|  |   +--rw operator-option-textstring! {operator-op-textstring}?
|  |   |  +--rw textstring* [textstring-id]
|  |   |     +--rw textstring-id    uint8
|  |   |     +--rw textstring       string
|  |   +--rw operator-option-var-data! {operator-op-var-data}?
|  |   |  +--rw int32val* [var-data-id]
|  |   |     +--rw var-data-id    uint8
|  |   |     +--rw var-data       binary
|  |   +--rw operator-option-dns-wire! {operator-op-dns-wire}?
|  |      +--rw operator-option-dns-wire* [operator-option-dns-wire-id]
|  |         +--rw operator-option-dns-wire-id    uint8
|  |         +--rw operator-option-dns-wire       binary
|  +--rw network-ranges
|  |  +--rw network-range* [network-range-id]
|  |   +--rw network-range-id       uint32
|  |   +--rw network-description    string
|  |   +--rw network-prefix         inet:ipv6-prefix
|  |   +--rw inherit-option-set     boolean
|  |   +--rw option-set-id?
|  |         -> /server/server-config/option-sets/option-set/id
|  |   +--rw reserved-addresses
|  |   |  +--rw static-binding* [cli-id]
|  |   |  |  +--rw cli-id         uint32
|  |   |  |  +--rw duid
|  |   |  |  |  +--rw type-code?                  uint16
```

```
|  |   |  |  |  +--rw (duid-type)?
|  |   |  |  |     +--:(duid-llt)
|  |   |  |  |     |  +--rw duid-llt-hardware-type?      uint16
|  |   |  |  |     |  +--rw duid-llt-time?      yang:timeticks
|  |   |  |  |     |  +--rw duid-llt-link-layer-addr? yang:mac-address
|  |   |  |  |     +--:(duid-en)
|  |   |  |  |     |  +--rw duid-en-enterprise-number?   uint32
|  |   |  |  |     |  +--rw duid-en-identifier?          string
|  |   |  |  |     +--:(duid-ll)
|  |   |  |  |     |  +--rw duid-ll-hardware-type?       uint16
|  |   |  |  |     |  +--rw duid-ll-link-layer-addr? yang:mac-address
|  |   |  |  |     +--:(duid-uuid)
|  |   |  |  |     |  +--rw uuid?                        yang:uuid
|  |   |  |  |     +--:(duid-invalid)
|  |   |  |  |        +--rw data?                        binary
|  |   |  |  +--rw reserv-addr*   inet:ipv6-address
|  |   |  +--rw other-reserv-addr*   inet:ipv6-address
|  |   +--rw reserved-prefixes
|  |   |  +--rw static-binding* [cli-id]
|  |   |  |  +--rw cli-id              uint32
|  |   |  |  +--rw duid
|  |   |  |  |  +--rw type-code?                  uint16
|  |   |  |  |  +--rw (duid-type)?
|  |   |  |  |     +--:(duid-llt)
|  |   |  |  |     |  +--rw duid-llt-hardware-type?      uint16
|  |   |  |  |     |  +--rw duid-llt-time?        yang:timeticks
|  |   |  |  |     |  +--rw duid-llt-link-layer-addr?  yang:mac-address
|  |   |  |  |     +--:(duid-en)
|  |   |  |  |     |  +--rw duid-en-enterprise-number?   uint32
|  |   |  |  |     |  +--rw duid-en-identifier?          string
|  |   |  |  |     +--:(duid-ll)
|  |   |  |  |     |  +--rw duid-ll-hardware-type?   uint16
|  |   |  |  |     |  +--rw duid-ll-link-layer-addr? yang:mac-address
|  |   |  |  |     +--:(duid-uuid)
|  |   |  |  |     |  +--rw uuid?                        yang:uuid
|  |   |  |  |     +--:(duid-invalid)
|  |   |  |  |        +--rw data?                        binary
|  |   |  |  +--rw reserv-prefix-len    uint8
|  |   |  |  +--rw reserv-prefix        inet:ipv6-prefix
|  |   |  +--rw exclude-prefix-len     uint8
|  |   |  +--rw exclude-prefix         inet:ipv6-prefix
|  |   |  +--rw other-reserv-prefix* [reserv-id]
|  |   |     +--rw reserv-id     uint32
|  |   |     +--rw prefix-len    uint8
|  |   |     +--rw prefix        inet:ipv6-prefix
|  |   +--rw address-pools
|  |   |  +--rw address-pool* [pool-id]
|  |   |     +--rw pool-id                          uint32
```

```
|  |   |     +--rw pool-prefi           inet:ipv6-prefix
|  |   |     +--rw start-address        inet:ipv6-address-no-zone
|  |   |     +--rw end-address          inet:ipv6-address-no-zone
|  |   |     +--rw renew-time                      yang:timeticks
|  |   |     +--rw rebind-time                     yang:timeticks
|  |   |     +--rw preferred-lifetime              yang:timeticks
|  |   |     +--rw valid-lifetime                  yang:timeticks
|  |   |     +--rw max-address-utilization-ratio   threshold
|  |   |     +--rw inherit-option-set              boolean
|  |   |     +--rw option-set-id
|  |   |        -> /server/server-config/option-sets/option-set/id
|  |   +--rw prefix-pools
|  |   |  +--rw prefix-pool* [pool-id]
|  |   |     +--rw pool-id                         uint32
|  |   |     +--rw prefix                          inet:ipv6-prefix
|  |   |     +--rw prefix-length                   uint8
|  |   |     +--rw renew-time                      yang:timeticks
|  |   |     +--rw rebind-time                     yang:timeticks
|  |   |     +--rw preferred-lifetime              yang:timeticks
|  |   |     +--rw valid-lifetime                  yang:timeticks
|  |   |     +--rw max-prefix-utilization-ratio    threshold
|  |   |     +--rw inherit-option-set              boolean
|  |   |     +--rw option-set-id?
|  |   |        -> /server/server-config/option-sets/option-set/id
|  |   +--rw hosts
|  |      +--rw host* [cli-id]
|  |         +--rw cli-id                 uint32
|  |         +--rw duid
|  |         |  +--rw type-code?                   uint16
|  |         |  +--rw (duid-type)?
|  |         |     +--:(duid-llt)
|  |         |     |  +--rw duid-llt-hardware-type?      uint16
|  |         |     |  +--rw duid-llt-time?        yang:timeticks
|  |         |     |  +--rw duid-llt-link-layer-addr?  yang:mac-address
|  |         |     +--:(duid-en)
|  |         |     |  +--rw duid-en-enterprise-number?  uint32
|  |         |     |  +--rw duid-en-identifier?         string
|  |         |     +--:(duid-ll)
|  |         |     |  +--rw duid-ll-hardware-type?     uint16
|  |         |     |  +--rw duid-ll-link-layer-addr?  yang:mac-address
|  |         |     +--:(duid-uuid)
|  |         |     |  +--rw uuid?                         yang:uuid
|  |         |     +--:(duid-invalid)
|  |         |        +--rw data?                         binary
|  |         +--rw inherit-option-set      boolean
|  |         +--rw option-set-id?
|  |             -> /server/server-config/option-sets/option-set/id
|  |         +--rw nis-domain-name?        string
```

```
|  |         +--rw nis-plus-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       uint32
|  |     |  +--rw subscriber-id    string
|  |     +--rw remote-host* [ent-num]
|  |        +--rw ent-num      uint32
|  |        +--rw remote-id    string
|  +--rw rsoo-enabled-options
|     +--rw rsoo-enabled-option* [option-code]
|        +--rw option-code    uint16
|        +--rw description    string
+--ro server-state
   +--ro network-ranges
   |  +--ro network-range* [network-range-id]
   |   +--ro network-range-id               uint32
   |   +--ro address-pools
   |   |  +--ro address-pool* [pool-id]
   |   |  |  +--ro pool-id                      uint32
   |   |  |  +--ro total-ipv6-count             uint64
   |   |  |  +--ro used-ipv6-count              uint64
   |   |  |  +--ro address-utilization-ratio    uint16
   |   |  +--ro binding-info* [cli-id]
   |   |     +--ro cli-id    uint32
   |   |     +--ro duid
   |   |     |  +--ro type-code?                   uint16
   |   |     |  +--ro (duid-type)?
   |   |     |     +--:(duid-llt)
   |   |     |     |  +--ro duid-llt-hardware-type?   uint16
   |   |     |     |  +--ro duid-llt-time?            yang:timeticks
   |   |     |     |  +--ro duid-llt-link-layer-addr? yang:mac-address
   |   |     |     +--:(duid-en)
   |   |     |     |  +--ro duid-en-enterprise-number?   uint32
   |   |     |     |  +--ro duid-en-identifier?          string
   |   |     |     +--:(duid-ll)
   |   |     |     |  +--ro duid-ll-hardware-type?       uint16
   |   |     |     |  +--ro duid-ll-link-layer-addr?  yang:mac-address
   |   |     |     +--:(duid-uuid)
   |   |     |     |  +--ro uuid?                        yang:uuid
   |   |     |     +--:(duid-invalid)
   |   |     |        +--ro data?                        binary
   |   |     +--ro cli-ia* [iaid]
   |   |        +--ro ia-type     string
```

```
|   |       +--ro iaid        uint32
|   |       +--ro cli-addr*   inet:ipv6-address
|   |       +--ro pool-id     uint32
|   +--ro prefix-pools
|   |  +--ro prefix-pool* [pool-id]
|   |  |  +--ro pool-id                    uint32
|   |  |  +--ro prefix-utilization-ratio   uint16
|   |  +--ro binding-info* [cli-id]
|   |     +--ro cli-id      uint32
|   |     +--ro duid
|   |     |  +--ro type-code?                  uint16
|   |     |  +--ro (duid-type)?
|   |     |     +--:(duid-llt)
|   |     |     |  +--ro duid-llt-hardware-type?   uint16
|   |     |     |  +--ro duid-llt-time?            yang:timeticks
|   |     |     |  +--ro duid-llt-link-layer-addr? yang:mac-address
|   |     |     +--:(duid-en)
|   |     |     |  +--ro duid-en-enterprise-number?   uint32
|   |     |     |  +--ro duid-en-identifier?          string
|   |     |     +--:(duid-ll)
|   |     |     |  +--ro duid-ll-hardware-type?    uint16
|   |     |     |  +--ro duid-ll-link-layer-addr?  yang:mac-address
|   |     |     +--:(duid-uuid)
|   |     |     |  +--ro uuid?                       yang:uuid
|   |     |     +--:(duid-invalid)
|   |     |        +--ro data?                       binary
|   |     +--ro cli-iapd* [iaid]
|   |        +--ro iaid             uint32
|   |        +--ro cli-prefix*      inet:ipv6-prefix
|   |        +--ro cli-prefix-len*  uint8
|   |        +--ro pool-id          uint32
|   +--ro address-prefix-assign-param* [cli-id]
|      +--ro cli-id                     uint32
|      +--ro source-ipv6-addr?          inet:ipv6-address
|      +--ro duid
|      |  +--ro type-code?                  uint16
|      |  +--ro (duid-type)?
|      |     +--:(duid-llt)
|      |     |  +--ro duid-llt-hardware-type?   uint16
|      |     |  +--ro duid-llt-time?            yang:timeticks
|      |     |  +--ro duid-llt-link-layer-addr? yang:mac-address
|      |     +--:(duid-en)
|      |     |  +--ro duid-en-enterprise-number?   uint32
|      |     |  +--ro duid-en-identifier?          string
|      |     +--:(duid-ll)
|      |     |  +--ro duid-ll-hardware-type?     uint16
|      |     |  +--ro duid-ll-link-layer-addr? yang:mac-address
|      |     +--:(duid-uuid)
```

```
    |      |     |  +--ro uuid?                          yang:uuid
    |      |     +--:(duid-invalid)
    |      |        +--ro data?                          binary
    |      +--ro iaid*                        uint32
    |      +--ro preferred-addr*              inet:ipv6-address
    |      +--ro preferred-prefix-len*        uint8
    |      +--ro client-fqdn?                 string
    |      +--ro client-link-layer-addr?      uint16
    |      +--ro client-enterprise-number?    uint32
    |      +--ro client-sys-archi-type*       uint16
    +--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

notifications:
+---n notifications
   +--ro dhcpv6-server-event
      +--ro pool-running-out
      |  +--ro max-address-utilization-ratio    uint16
      |  +--ro address-utilization-ratio        uint16
      |  +--ro max-prefix-utilization-ratio     uint16
      |  +--ro prefix-utilization-ratio         uint16
      |  +--ro duid
      |  |  +--ro type-code?                   uint16
      |  |  +--ro (duid-type)?
      |  |     +--:(duid-llt)
      |  |     |  +--ro duid-llt-hardware-type?    uint16
      |  |     |  +--ro duid-llt-time?             yang:timeticks
      |  |     |  +--ro duid-llt-link-layer-addr?  yang:mac-address
      |  |     +--:(duid-en)
      |  |     |  +--ro duid-en-enterprise-number? uint32
      |  |     |  +--ro duid-en-identifier?        string
      |  |     +--:(duid-ll)
      |  |     |  +--ro duid-ll-hardware-type?    uint16
      |  |     |  +--ro duid-ll-link-layer-addr?  yang:mac-address
      |  |     +--:(duid-uuid)
      |  |     |  +--ro uuid?                     yang:uuid
```

```
      |  |     +--:(duid-invalid)
      |  |        +--ro data?                          binary
      |  +--ro serv-name?                         string
      |  +--ro pool-name                          string
      +--ro invalid-client-detected
         +--ro duid
         |  +--ro type-code?                    uint16
         |  +--ro (duid-type)?
         |     +--:(duid-llt)
         |     |  +--ro duid-llt-hardware-type?    uint16
         |     |  +--ro duid-llt-time?            yang:timeticks
         |     |  +--ro duid-llt-link-layer-addr? yang:mac-address
         |     +--:(duid-en)
         |     |  +--ro duid-en-enterprise-number?  uint32
         |     |  +--ro duid-en-identifier?         string
         |     +--:(duid-ll)
         |     |  +--ro duid-ll-hardware-type?    uint16
         |     |  +--ro duid-ll-link-layer-addr?  yang:mac-address
         |     +--:(duid-uuid)
         |     |  +--ro uuid?                     yang:uuid
         |     +--:(duid-invalid)
         |        +--ro data?                     binary
         +--ro description?   string
```

Figure 1: DHCPv6 Data Model Structure

Introduction of important nodes:

- server-config: This container contains the configuration data of a server.

- 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.

- 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 bytes) content field. Currently there are four defined types of DUIDs in [RFC3315] and [RFC6355] - DUID-LLT, DUID-EN, DUID-LL and DUID-UUID. DUID-INVALID represents those unconventional DUIDs.

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

- operator-option-ipv6-address, operator-option-single-flag, operator-option-ipv6-prefix, operator-option-int32, operator-option-int16, operator-option-int8, operator-option-uri, operator-option-textstring, operator-option-var-data, operator-option-dns-wire: are generic option formats described in [RFC7227].

- stateless-service: A boolean value specifies whether the server support client-server exchanges involving two messages defined in [RFC3315].

- 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.

- interfaces-config: A leaf list to denote which one or more interfaces the server should listen on. The default value is to listen on all the interfaces. This node is also used to set a unicast address for the server to listen with a specific interface. For example, if the server is being configured to listen on a unicast address assigned to a specific interface, the format "eth1/2001:db8::1" can be used.

- option-sets: DHCPv6 employs various options to carry additional information and parameters in DHCP messages. This container defines all the possible options that need to be configured at the server side. The relevant RFCs that define those options include: [RFC3315], [RFC3319], [RFC3646], [RFC3898], [RFC4242], [RFC4704], [RFC4833], [RFC5908], [RFC5970], [RFC4075], [RFC6334], [RFC6784], [RFC7078], [RFC7083], [RFC7291], [RFC7598].

- option-set: A server may allow different option sets to be configured for different conditions (i.e. different networks, clients and etc). This "option-set" list enables various sets of options being defined and configured in a single server. Different sets are distinguished by the key called "option-set-id". All the possible options discussed above are defined in the list and each option is corresponding to a container. Since all the options in the list are optional, each container in this list has a 'presence' statement to indicate whether this option (container) will be included in the current option set or not. In addition, each container also has a 'if-feature' statement to indicate whether the server supports this option (container).

- network-ranges: This model supports a hierarchy to achieve dynamic configuration. That is to say we could configure the server at different levels through this model. The top level is a global level which is defined as the container "network-ranges". The following levels are defined as sub-containers under it. The

"network-ranges" contains the parameters (e.g. option-sets) that would be allocated to all the clients served by this server

- network-range: Under the "network-ranges" container, a "network-range" list is defined to configure the server at a network level which is also considered as the second level. Different network are identified by the key "network-range-id". This is because a server may have different configuration parameters (e.g. option sets) for different networks.

- address-pools: Under the "network-range" list, a container describes the DHCPv6 server's address pools for a specific network is defined. This container supports the server to be configured at a pool level.

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

- max-address-utilization-ratio: The threshold of address pool utiliztion, the value of which is settable.

- binding-info: 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 under the "network-range" list will be valid to define the delegating router's prefix pools for a specific network. This container also supports the server to be configured at a pool level.

- 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".

- max-prefix-utilization-ratio: The threshold of prefix pool utiliztion, the value of which is settable.

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

- hosts: A server may also desire to be configured at a host level under some circumstances. This container include a list called "host" to allow the server carrying different parameters (e.g. option sets) for different hosts.

- 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.

- rsoo-enabled-options: [RFC6422] requires that the server SHOULD have an administrator-configurable list of RSOO-enabled options. This container include a list called "rsoo-enabled-option" to allow new RSOO-enabled options to be defined at the server side.

- server-state: This container includes the state data of a server.

- address-prefix-assign-param: This list includes some parameters/identifiers that the server obtains from DHCPv6 options in this network-range. The server may take these parameters/identifiers into account when assigning a(n) address/prefix.

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

Information about notifications:

- pool-running-out: raised when the address/prefix pool is going to run out. A threshold for utilization ratio of the pool has been defined in the server feature so that it will notify the administrator when the utilization ratio reaches the threshold, and such threshold is a settable parameter.

- 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.

2.2. DHCPv6 Relay Tree Diagrams

```
module: ietf-dhcpv6-relay
    +--rw relay!
       +--rw relay-config
       |  +--rw relay-attributes
       |  |  +--rw name?          string
       |  |  +--rw description?   string
       |  |  +--rw dest-addrs*    inet:ipv6-address
       |  |  +--rw subscribers* [subscriber]
       |  |  |  +--rw subscriber       uint8
       |  |  |  +--rw subscriber-id    string
       |  |  +--rw remote-host* [ent-num]
       |  |  |  +--rw ent-num      uint32
       |  |  |  +--rw remote-id    string
       |  |  +--rw vendor-info
       |  |     +--rw ent-num    uint32
       |  |     +--rw data*      string
```

```
|  +--rw rsoo-option-sets
|  |  +--rw option-set* [id]
|  |     +--rw id                              uint32
|  |     +--rw erp-local-domain-name-option!
                            {erp-local-domain-name-op}?
|  |        +--rw erp-for-client* [cli-id]
|  |           +--rw cli-id      uint32
|  |           +--rw duid
|  |           |  +--rw type-code?                  uint16
|  |           |  +--rw (duid-type)?
|  |           |     +--:(duid-llt)
|  |           |     |  +--rw duid-llt-hardware-type?      uint16
|  |           |     |  +--rw duid-llt-time?
                                                 yang:timeticks
|  |           |     |  +--rw duid-llt-link-layer-addr?
                                               yang:mac-address
|  |           |     +--:(duid-en)
|  |           |     |  +--rw duid-en-enterprise-number?   uint32
|  |           |     |  +--rw duid-en-identifier?          string
|  |           |     +--:(duid-ll)
|  |           |     |  +--rw duid-ll-hardware-type?       uint16
|  |           |     |  +--rw duid-ll-link-layer-addr?
                                               yang:mac-address
|  |           |     +--:(duid-uuid)
|  |           |     |  +--rw uuid?                        yang:uuid
|  |           |     +--:(duid-invalid)
|  |           |        +--rw data?                        binary
|  |           +--rw erp-name    string
|  +--rw relay-if* [if-name]
|     +--rw if-name               string
|     +--rw enable                boolean
|     +--rw ipv6-address?         inet:ipv6-address
|     +--rw interface-id?         string
|     +--rw rsoo-option-set-id?
              -> /relay/relay-config/rsoo-option-sets/option-set/id
|     +--rw next-entity* [dest-addr]
|        +--rw dest-addr    inet:ipv6-address
|        +--rw available    boolean
|        +--rw multicast    boolean
|        +--rw server       boolean
+--ro relay-state
   +--ro relay-if* [if-name]
   |  +--ro if-name        string
   |  +--ro pd-route* [pd-route-id]
   |  |  +--ro pd-route-id             uint8
   |  |  +--ro requesting-router-id    uint32
   |  |  +--ro delegating-router-id    uint32
   |  |  +--ro next-router             inet:ipv6-address
```

```
          |  |  +--ro last-router             inet:ipv6-address
          |  +--ro next-entity* [dest-addr]
          |     +--ro dest-addr       inet:ipv6-address
          |     +--ro packet-stats
          |        +--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

  notifications:
    +---n notifications
       +--ro dhcpv6-relay-event
          +--ro topo-changed
             +--ro relay-if-name       string
             +--ro first-hop           boolean
             +--ro last-entity-addr    inet:ipv6-address
```

Introduction of important nodes:

- relay-config: This container contains the configuration data of the relay.

- 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.

- 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.

- rsoo-options-sets: DHCPv6 relay agent could provide some information that would be useful to DHCPv6 client. Since relay agent cannot provide options directly to the client, [RFC6422] defines RSOO-enabled options to propose options for the server to send to the client. This container models such RSOO-enabled options.

- option-set: This list under the "rsoo-option-sets" container is similar to the that defined in server module. It allows the relay to implement several sets of RSOO-enabled options for different interfaces. The list only include the EAP Re-authentication Protocol (ERP) Local Domain Name DHCPv6 Option defined in [RFC6440], since it is the only one RSOO-enabled options accepted by IANA so far.

- relay-if: A relay agent may have several interfaces, we should provide a way to configure and manage parameters on the interface-level. A list that describes specific interfaces and their corresponding parameters is employed to fulfil the configfuration. Here we use a string called "if-name" as the key of list.

- relay-state: This container contains the configuration data of the relay.

- pd-route: A sub-container of "relay-if" which describes the route for delegated prefixes into the provider edge router.

- 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.

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

- relay-stats: The "relay-stats" container records and presents the overall packet statistics of the relay agent.

Information about notifications:

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

2.3. DHCPv6 Client Tree Diagrams

```
module: ietf-dhcpv6-client
    +--rw client!
       +--rw client-config
       |  +--rw duid
       |  |  +--rw type-code?                  uint16
       |  |  +--rw (duid-type)?
       |  |     +--:(duid-llt)
       |  |     |  +--rw duid-llt-hardware-type?      uint16
       |  |     |  +--rw duid-llt-time?               yang:timeticks
       |  |     |  +--rw duid-llt-link-layer-addr?    yang:mac-address
       |  |     +--:(duid-en)
       |  |     |  +--rw duid-en-enterprise-number?   uint32
       |  |     |  +--rw duid-en-identifier?          string
       |  |     +--:(duid-ll)
       |  |     |  +--rw duid-ll-hardware-type?       uint16
       |  |     |  +--rw duid-ll-link-layer-addr?     yang:mac-address
       |  |     +--:(duid-uuid)
       |  |     |  +--rw uuid?                        yang:uuid
       |  |     +--:(duid-invalid)
       |  |        +--rw data?                        binary
       |  +--rw client-if* [if-name]
       |     +--rw if-name                      string
       |     +--rw cli-id                       uint32
       |     +--rw description?                 string
       |     +--rw pd-function                  boolean
       |     +--rw rapid-commit                 boolean
       |     +--rw mo-tab
       |     |  +--rw m-tab    boolean
       |     |  +--rw o-tab    boolean
       |     +--rw client-configured-options
       |        +--rw new-or-standard-cli-option* [option-code]
       |        |  +--rw option-code           uint16
       |        |  +--rw option-name           string
       |        |  +--rw option-description    string
       |        |  +--rw option-reference?     string
       |        |  +--rw option-value          string
       |        +--rw option-request-option! {option-request-op}?
       |        |  +--rw oro-option* [option-code]
       |        |     +--rw option-code    uint16
       |        |     +--rw description    string
       |        +--rw user-class-option! {user-class-op}?
       |        |  +--rw user-class* [user-class-id]
       |        |     +--rw user-class-id      uint8
       |        |     +--rw user-class-data    string
       |        +--rw vendor-class-option! {vendor-class-op}?
       |        |  +--rw enterprise-number    uint32
```

```
|       |  +--rw vendor-class* [vendor-class-id]
|       |     +--rw vendor-class-id      uint8
|       |     +--rw vendor-class-data    string
|       +--rw client-fqdn-option! {client-fqdn-op}?
|       |  +--rw fqdn                      string
|       |  +--rw server-initiate-update    boolean
|       |  +--rw client-initiate-update    boolean
|       +--rw client-arch-type-option! {client-arch-type-op}?
|       |  +--rw architecture-types* [type-id]
|       |     +--rw type-id          uint16
|       |     +--rw most-preferred   boolean
|       +--rw client-network-interface-identifier-option!
                    {client-network-interface-identifier-op}?
|       |  +--rw type     uint8
|       |  +--rw major    uint8
|       |  +--rw minor    uint8
|       +--rw kbr-principal-name-option! {kbr-principal-name-op}?
|       |  +--rw principle-name* [principle-name-id]
|       |     +--rw principle-name-id    uint8
|       |     +--rw name-type            int32
|       |     +--rw name-string          string
|       +--rw kbr-realm-name-option! {kbr-realm-name-op}?
|       |  +--rw realm-name    string
|       +--rw client-link-layer-addr-option!
                          {client-link-layer-addr-op}?
|          +--rw link-layer-type    uint16
|          +--rw link-layer-addr    string
+--ro client-state
   +--ro if-other-paras
      +--ro server-unicast-option! {server-unicast-op}?
      |  +--ro server-address?   inet:ipv6-address
      +--ro sip-server-domain-name-list-option!
                      {sip-server-domain-name-list-op}?
      |  +--ro sip-serv-domain-name    string
      +--ro sip-server-address-list-option!
                          {sip-server-address-list-op}?
      |  +--ro sip-server* [sip-serv-id]
      |     +--ro sip-serv-id      uint8
      |     +--ro sip-serv-addr    inet:ipv6-address
      +--ro dns-config-option! {dns-config-op}?
      |  +--ro dns-server* [dns-serv-id]
      |     +--ro dns-serv-id      uint8
      |     +--ro dns-serv-addr    inet:ipv6-address
      +--ro domain-searchlist-option! {domain-searchlist-op}?
      |  +--ro domain-searchlist* [domain-searchlist-id]
      |     +--ro domain-searchlist-id        uint8
      |     +--ro domain-search-list-entry    string
      +--ro nis-config-option! {nis-config-op}?
```

```
|  +--ro nis-server* [nis-serv-id]
|     +--ro nis-serv-id      uint8
|     +--ro nis-serv-addr    inet:ipv6-address
+--ro nis-plus-config-option! {nis-plus-config-op}?
|  +--ro nis-plus-server* [nis-plus-serv-id]
|     +--ro nis-plus-serv-id      uint8
|     +--ro nis-plus-serv-addr    inet:ipv6-address
+--ro nis-domain-name-option! {nis-domain-name-op}?
|  +--ro nis-domain-name?   string
+--ro nis-plus-domain-name-option! {nis-plus-domain-name-op}?
|  +--ro nis-plus-domain-name?   string
+--ro sntp-server-option! {sntp-server-op}?
|  +--ro sntp-server* [sntp-serv-id]
|     +--ro sntp-serv-id      uint8
|     +--ro sntp-serv-addr    inet:ipv6-address
+--ro info-refresh-time-option! {info-refresh-time-op}?
|  +--ro info-refresh-time    yang:timeticks
+--ro client-fqdn-option! {client-fqdn-op}?
|  +--ro server-initiate-update    boolean
|  +--ro client-initiate-update    boolean
|  +--ro modify-name-from-cli      boolean
+--ro posix-timezone-option! {posix-timezone-op}?
|  +--ro tz-posix    string
+--ro tzdb-timezone-option! {tzdb-timezone-op}?
|  +--ro tz-database    string
+--ro ntp-server-option! {ntp-server-op}?
|  +--ro ntp-server* [ntp-serv-id]
|     +--ro ntp-serv-id                    uint8
|     +--ro (ntp-time-source-suboption)?
|        +--:(server-address)
|        |  +--ro ntp-serv-addr-suboption*  inet:ipv6-address
|        +--:(server-multicast-address)
|        |  +--ro ntp-serv-mul-addr-suboption*
                                           inet:ipv6-address
|        +--:(server-fqdn)
|           +--ro ntp-serv-fqdn-suboption*       string
+--ro boot-file-url-option! {boot-file-url-op}?
|  +--ro boot-file* [boot-file-id]
|     +--ro boot-file-id          uint8
|     +--ro suitable-arch-type*   uint16
|     +--ro suitable-net-if*      uint32
|     +--ro boot-file-url         string
+--ro boot-file-param-option! {boot-file-param-op}?
|  +--ro boot-file-paras* [para-id]
|     +--ro para-id      uint8
|     +--ro parameter    string
+--ro aftr-name-option! {aftr-name-op}?
|  +--ro tunnel-endpoint-name    string
```

```
        +--ro kbr-default-name-option! {kbr-default-name-op}?
        |  +--ro default-realm-name    string
        +--ro kbr-kdc-option! {kbr-kdc-op}?
        |  +--ro kdc-info* [kdc-id]
        |     +--ro kdc-id            uint8
        |     +--ro priority          uint16
        |     +--ro weight            uint16
        |     +--ro transport-type    uint8
        |     +--ro port-number       uint16
        |     +--ro kdc-ipv6-addr     inet:ipv6-address
        |     +--ro realm-name        string
        +--ro sol-max-rt-option! {sol-max-rt-op}?
        |  +--ro sol-max-rt-value    yang:timeticks
        +--ro inf-max-rt-option! {inf-max-rt-op}?
        |  +--ro inf-max-rt-value    yang:timeticks
        +--ro addr-selection-option! {addr-selection-op}?
        |  +--ro a-bit-set       boolean
        |  +--ro p-bit-set       boolean
        |  +--ro policy-table* [policy-id]
        |     +--ro policy-id     uint8
        |     +--ro label         uint8
        |     +--ro precedence    uint8
        |     +--ro prefix-len    uint8
        |     +--ro prefix        inet:ipv6-prefix
        +--ro pcp-server-option! {pcp-server-op}?
        |  +--ro pcp-server* [pcp-serv-id]
        |     +--ro pcp-serv-id      uint8
        |     +--ro pcp-serv-addr    inet:ipv6-address
        +--ro s46-rule-option! {s46-rule-op}?
        |  +--ro s46-rule* [rule-id]
        |     +--ro rule-id           uint8
        |     +--ro rule-type         enumeration
        |     +--ro prefix4-len       uint8
        |     +--ro ipv4-prefix       inet:ipv4-prefix
        |     +--ro prefix6-len       uint8
        |     +--ro ipv6-prefix       inet:ipv6-prefix
        |     +--ro port-parameter
        |        +--ro offset      uint8
        |        +--ro psid-len    uint8
        |        +--ro psid        uint16
        +--ro s46-br-option! {s46-br-op}?
        |  +--ro br* [br-id]
        |     +--ro br-id           uint8
        |     +--ro br-ipv6-addr    inet:ipv6-address
        +--ro s46-dmr-option! {s46-dmr-op}?
        |  +--ro dmr* [dmr-id]
        |     +--ro dmr-id             uint8
        |     +--ro dmr-prefix-len     uint8
```

```
            |     +--ro dmr-ipv6-prefix    inet:ipv6-prefix
            +--ro s46-v4-v6-binding-option! {s46-v4-v6-binding-op}?
               +--ro ce* [ce-id]
                  +--ro ce-id               uint8
                  +--ro ipv4-addr           inet:ipv4-address
                  +--ro bind-prefix6-len    uint8
                  +--ro bind-ipv6-prefix    inet:ipv6-prefix
                  +--ro port-parameter
                     +--ro offset      uint8
                     +--ro psid-len    uint8
                     +--ro psid        uint16

  notifications:
    +---n notifications
       +--ro dhcpv6-client-event
          +--ro ia-lease-event
          |  +--ro event-type     enumeration
          |  +--ro duid
          |  |  +--ro type-code?                   uint16
          |  |  +--ro (duid-type)?
          |  |     +--:(duid-llt)
          |  |     |  +--ro duid-llt-hardware-type?      uint16
          |  |     |  +--ro duid-llt-time?               yang:timeticks
          |  |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
          |  |     +--:(duid-en)
          |  |     |  +--ro duid-en-enterprise-number?   uint32
          |  |     |  +--ro duid-en-identifier?          string
          |  |     +--:(duid-ll)
          |  |     |  +--ro duid-ll-hardware-type?       uint16
          |  |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
          |  |     +--:(duid-uuid)
          |  |     |  +--ro uuid?                        yang:uuid
          |  |     +--:(duid-invalid)
          |  |        +--ro data?                        binary
          |  +--ro iaid           uint32
          |  +--ro serv-name?     string
          |  +--ro description?   string
          +--ro invalid-ia-detected
          |  +--ro duid
          |  |  +--ro type-code?                   uint16
          |  |  +--ro (duid-type)?
          |  |     +--:(duid-llt)
          |  |     |  +--ro duid-llt-hardware-type?      uint16
          |  |     |  +--ro duid-llt-time?               yang:timeticks
          |  |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
          |  |     +--:(duid-en)
          |  |     |  +--ro duid-en-enterprise-number?   uint32
          |  |     |  +--ro duid-en-identifier?          string
```

```
|  |     +--:(duid-ll)
|  |     |  +--ro duid-ll-hardware-type?       uint16
|  |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
|  |     +--:(duid-uuid)
|  |     |  +--ro uuid?                        yang:uuid
|  |     +--:(duid-invalid)
|  |        +--ro data?                        binary
|  +--ro cli-duid       uint32
|  +--ro iaid           uint32
|  +--ro serv-name?     string
|  +--ro description?   string
+--ro retransmission-failed
|  +--ro duid
|  |  +--ro type-code?                  uint16
|  |  +--ro (duid-type)?
|  |     +--:(duid-llt)
|  |     |  +--ro duid-llt-hardware-type?      uint16
|  |     |  +--ro duid-llt-time?               yang:timeticks
|  |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
|  |     +--:(duid-en)
|  |     |  +--ro duid-en-enterprise-number?   uint32
|  |     |  +--ro duid-en-identifier?          string
|  |     +--:(duid-ll)
|  |     |  +--ro duid-ll-hardware-type?       uint16
|  |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
|  |     +--:(duid-uuid)
|  |     |  +--ro uuid?                        yang:uuid
|  |     +--:(duid-invalid)
|  |        +--ro data?                        binary
|  +--ro description    enumeration
+--ro failed-status-turn-up
   +--ro duid
   |  +--ro type-code?                  uint16
   |  +--ro (duid-type)?
   |     +--:(duid-llt)
   |     |  +--ro duid-llt-hardware-type?      uint16
   |     |  +--ro duid-llt-time?               yang:timeticks
   |     |  +--ro duid-llt-link-layer-addr?    yang:mac-address
   |     +--:(duid-en)
   |     |  +--ro duid-en-enterprise-number?   uint32
   |     |  +--ro duid-en-identifier?          string
   |     +--:(duid-ll)
   |     |  +--ro duid-ll-hardware-type?       uint16
   |     |  +--ro duid-ll-link-layer-addr?     yang:mac-address
   |     +--:(duid-uuid)
   |     |  +--ro uuid?                        yang:uuid
   |     +--:(duid-invalid)
   |        +--ro data?                        binary
```

```
            +--ro status-code    enumeration
```

Introduction of important nodes:

- client-config: This container includes the configuration data of the client.

- client-if: A client may have several interfaces, it is more reasonable to configure and manage parameters on the interface-level. 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. This leaf are same as the "duid" leaf in "dhcpv6-server" feature.

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

- 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.

- 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.

- client-configured-options: Similar to the server, the client also need to configure some options to fulfil some desired functions. This container include all the potential options that need to be configured at the client side. The relevant RFCs that define those options include: [RFC3315], [RFC4704], [RFC5970], [RFC6784], [RFC6939].

- option-request-option: This container provide a way to configure the list of options that the client will request in its ORO option.

- client-state: This container includes the state data of the client.

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

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

Information about notifications:

- ia-lease-event: raised when the client was allocated a new IA from the server or it renew/rebind/release its current IA.

- 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.

3. DHCPv6 YANG Model

3.1. DHCPv6 Server YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-server@2017-12-22.yang"
module ietf-dhcpv6-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server";
  prefix "dhcpv6-server";

  import ietf-inet-types {
    prefix inet;
    revision-date "2013-07-15";
  }
  import ietf-yang-types {
    prefix yang;
    revision-date "2013-07-15";
  }
  import ietf-dhcpv6-options {
```

```
    prefix dhcpv6-options;
    revision-date "2017-12-22";
  }

  organization "DHC WG";
  contact "yong@csnet1.cs.tsinghua.edu.cn
    lh.sunlinh@gmail.com
    ian.farrer@telekom.de
    sladjana.zechlin@telekom.de
    hezihao9512@gmail.com";

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

  revision 2017-12-22 {
    description "Resolve most issues on Ian's github.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  revision 2017-11-24 {
    description "First version of the separated server specific
      YANG model.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  /*
   * Typedef
   */

  typedef threshold {
    type union {
      type uint16 {
        range 0..100;
      }
      type enumeration {
        enum "disabled" {
          description "No threshold";
        }
      }
    }
    description "Threshold value in percent";
  }

  /*
   * Grouping
   */

  grouping vendor-infor {
```

```
   description "Vendor information.";
   container vendor-info {
     description "";
     leaf ent-num {
       type uint32;
       mandatory true;
       description "enterprise number";
     }
     leaf-list data {
       type string;
       description "specific vendor info";
     }
   }
  }

  grouping duid {
      description "DHCP Unique Identifier";
      reference "RFC3315: Section 9 and RFC6355: Section 4";
      leaf type-code {
        type uint16;
        default 65535;
        description "Type code of this DUID";
      }
      choice duid-type {
      default duid-invalid;
        description "Selects the format for the DUID.";
        case duid-llt {
          description "DUID Based on Link-layer Address Plus Time (Type 1 -
DUID-LLT)";
          reference "RFC3315 Section 9.2";
          leaf duid-llt-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
          leaf duid-llt-time {
            type yang:timeticks;
            description "The time value is the time that the DUID is generated
              represented in seconds since midnight (UTC), January 1, 2000,
              modulo 2^32.";
          }
          leaf duid-llt-link-layer-addr {
            type yang:mac-address;
            description "Link-layer address as described in RFC2464";
          }
        }
        case duid-en {
          description "DUID Assigned by Vendor Based on Enterprise Number (Type
2 - DUID-EN)";
```

```
          reference "RFC3315 Section 9.3";
          leaf duid-en-enterprise-number {
```

```
            type uint32;
            description "Vendor's registered Private Enterprise Number as
              maintained by IANA";
          }
          leaf duid-en-identifier {
            type string;
            description "Indentifier, unique to the device that is using it";
          }
        }
        case duid-ll {
          description "DUID Based on Link-layer Address (Type 3 - DUID-LL)";
          reference "RFC3315 Section 9.4";
          leaf duid-ll-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
          leaf duid-ll-link-layer-addr {
              type yang:mac-address;
              description "Link-layer address as described in RFC2464";
        }
        }
        case duid-uuid {
          description "DUID Based on Universally Unique Identifier (Type 4 -
DUID-UUID)";
          reference "RFC6335 Defination of the UUID-Based Unique Identifier";
          leaf uuid {
            type yang:uuid;
            description "A Universally Unique IDentifier in the string
representation
                defined in RFC 4122.  The canonical representation uses
                lowercase characters";
          }
        }
        case duid-invalid {
          description "DUID based on free raw bytes";
          leaf data {
            type binary;
            description "The bits to be used as the identifier";
          }
        }
      }
    }

  /*
   * Data Nodes
   */

  container server {
```

```
    presence "Enables the server";
    description "DHCPv6 server portion";
```

```
    /*
     * Configuration data
     */
    container server-config {
      description "configuration tree of server";
         container serv-attributes {
            description "This container contains basic attributes
              of a DHCPv6 server such as IPv6 address, server name
              and so on. Some optional functions that can be provided
              by 0the server is also included.";
            leaf name {
              type string;
              description "server's name";
            }
            container duid {
              description "Sets the DUID";
              uses duid;
            }
            leaf-list ipv6-address {
              type inet:ipv6-address;
              description "Server's IPv6 address.";
            }
            leaf description {
              type string;
              description "Description of the server.";
            }
            leaf pd-function {
              type boolean;
              mandatory true;
              description "Whether the server can act as a
                delegating router to perform prefix delegation
                ([RFC3633]).";
            }
            leaf stateless-service {
              type boolean;
              mandatory true;
              description "A boolean value specifies whether
                the server support client-server exchanges
                involving two messages defined in ([RFC3315]).";
            }
            leaf rapid-commit {
              type boolean;
              mandatory true;
              description "A boolean value specifies whether
                the server support client-server exchanges
                involving two messages defined in ([RFC3315]).";
            }
            leaf-list interfaces-config {
```

```
              // Note - this should probably be references to
              // entries in the ietf-interfaces model
              type string;
              description "A leaf list to denote which one or
                more interfaces the server should listen on. The
                default value is to listen on all the interfaces.
                This node is also used to set a unicast address
                for the server to listen with a specific interface.
                  For example, if people want the server to listen
                    on a unicast address with a specific interface, he
                    can use the format like 'eth1/2001:db8::1'.";
            }
            uses vendor-infor;
          }

          container option-sets {
            description "DHCPv6 employs various options to carry additional
                  information and parameters in DHCP messages. This container
defines
                  all the possible options that need to be configured at the
server
                  side. ";
                list option-set {
                  key id;
                description "A server may allow different option sets to be
                    configured for different conditions (i.e. different
networks,
                    clients and etc). This 'option-set' list enables various
sets of
                    options being defined and configured in a single server.
Different
                    sets are distinguished by the key called 'option-set-id'.
All the
                    possible options discussed above are defined in the list
and each
                    option is corresponding to a container. Since all the
options in
                    the list are optional, each container in this list has a
'presence'
                    statement to indicate whether this option (container) will
be
                    included in the current option set or not. In addition,
each container
                    also has a 'if-feature' statement to indicate whether the
server
                    supports this option (container).";
                  leaf id {
                    type uint32;
```

```
          description "option set id";
        }
      uses dhcpv6-options:server-option-definitions;
      uses dhcpv6-options:custom-option-definitions;
    }
  }

  container network-ranges {
    description "This model supports a hierarchy
      to achieve dynamic configuration. That is to
      say we could configure the server at different
      levels through this model. The top level is a
```

```
        global level which is defined as the container
        'network-ranges'. The following levels are
        defined as sub-containers under it. The
        'network-ranges' contains the parameters
        (e.g. option-sets) that would be allocated to
        all the clients served by this server.";
      list network-range {
        key network-range-id;
        description "Under the 'network-ranges'
          container, a 'network-range' list is
          defined to configure the server at a
          network level which is also considered
          as the second level. Different network
          are identified by the key 'network-range-id'.
          This is because a server may have different
          configuration parameters (e.g. option sets)
          for different networks.";
        leaf network-range-id {
          type uint32;
          mandatory true;
          description "equivalent to subnet id";
        }
        leaf network-description {
          type string;
          mandatory true;
          description "description of the subnet";
        }
        leaf network-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "subnet prefix";
        }
        leaf inherit-option-set {
          type boolean;
          mandatory true;
          description "indicate whether to inherit
            the configuration from higher level";
        }
        leaf option-set-id {
          type leafref {
            path "/server/server-config/option-sets/option-set/id";

          }
          description "The ID field of relevant option-set to be
            provisioned to clients of this network-range.";
        }

        container reserved-addresses {
```

```
      description "reserved addresses";
      list static-binding {
        key cli-id;
        description "static binding of
          reserved addresses";
        leaf cli-id {
          type uint32;
          mandatory true;
          description "client id";
        }
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf-list reserv-addr {
          type inet:ipv6-address;
          description "reserved addr";
        }
      }
      leaf-list other-reserv-addr {
        type inet:ipv6-address;
        description "other reserved
          addr";
      }
    }

    container reserved-prefixes {
      description "reserved prefixes";
      list static-binding {
        key cli-id;
        description "static binding";
        leaf cli-id {
          type uint32;
          mandatory true;
          description "client id";
        }
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf reserv-prefix-len {
          type uint8;
          mandatory true;
          description "reserved
            prefix length";
        }
        leaf reserv-prefix {
          type inet:ipv6-prefix;
```

```
            mandatory true;
            description
              "reserved prefix";
          }
        }
        leaf exclude-prefix-len {
          type uint8;
          mandatory true;
          description "exclude prefix
            length";
        }
        leaf exclude-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "exclude prefix";
        }
        list other-reserv-prefix {
          key reserv-id;
          description
            "other reserved prefix";
          leaf reserv-id {
            type uint32;
            mandatory true;
            description
              "reserved prefix id";
          }
          leaf prefix-len {
            type uint8;
            mandatory true;
            description "prefix length";
          }
          leaf prefix {
            type inet:ipv6-prefix;
            mandatory true;
            description
              "reserved prefix";
          }
        }
      }

      container address-pools {
        description "A container that describes
          the DHCPv6 server's address pools.";
        list address-pool {
          key pool-id;
          description "A DHCPv6 server can
            be configured with several address
            pools. This list defines such
```

```
        address pools which are distinguished
        by the key called 'pool-name'.";
      leaf pool-id {
        type uint32;
        mandatory true;
        description "pool id";
      }
      leaf pool-prefix {
        type inet:ipv6-prefix;
        mandatory true;
        description "pool prefix";
      }
      leaf start-address {
        type inet:ipv6-address-no-zone;
        mandatory true;
        description "start address";
      }
      leaf end-address {
        type inet:ipv6-address-no-zone;
        mandatory true;
        description "end address";
      }
      leaf renew-time {
        type yang:timeticks;
        mandatory true;
        description "renew time";
      }
      leaf rebind-time {
        type yang:timeticks;
        mandatory true;
        description "rebind time";
      }
      leaf preferred-lifetime {
        type yang:timeticks;
        mandatory true;
        description "preferred lifetime
          for IA";
      }
      leaf valid-lifetime {
        type yang:timeticks;
        mandatory true;
        description "valid liftime for IA";
      }
      leaf max-address-utilization-ratio {
        type threshold;
        mandatory true;
        description "address pool utilization ratio threshold";
      }
```

```
                    leaf inherit-option-set {
                      type boolean;
                      mandatory true;
                      description "indicate whether to
                        inherit the configuration from
                        higher level";
                    }
                    leaf option-set-id {
                      type leafref {
                        path "/server/server-config/option-sets/option-set/id";
                      }
                      mandatory true;
                      description "The ID field of relevant option-set to be
                        provisioned to clients of this address-pool.";
                    }
                  }
                }

                  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 {
                      key pool-id;
                      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'.";
                      leaf pool-id {
                        type uint32;
                        mandatory true;
                        description "pool id";
                      }
                      leaf prefix {
                        type inet:ipv6-prefix;
                        mandatory true;
                        description "ipv6 prefix";
                      }
                      leaf prefix-length {
                        type uint8;
                        mandatory true;
                        description "prefix length";
                      }
                      leaf renew-time {
                        type yang:timeticks;
                        mandatory true;
```

```
                        description "renew time";
                      }
                      leaf rebind-time {
                        type yang:timeticks;
                        mandatory true;
                        description "rebind time";
                      }
                      leaf preferred-lifetime {
                        type yang:timeticks;
                        mandatory true;
                        description "preferred lifetime for
                          IA";
                      }
                      leaf valid-lifetime {
                        type yang:timeticks;
                        mandatory true;
                        description "valid lifetime for IA";
                      }
                      leaf max-prefix-utilization-ratio {
                        type threshold;
                        mandatory true;
                        description "prefix pool utilization ratio threshold";
                      }
                      leaf inherit-option-set {
                        type boolean;
                        mandatory true;
                        description "whether to inherit
                          configuration from higher level";
                      }
                      leaf option-set-id {
                        type leafref {
                          path "/server/server-config/option-sets/option-set/
id";
                        }
                        description "The ID field of relevant option-set to be
                          provisioned to clients of this prefix-pool.";
                      }
                    }
                  }
                  container hosts {
                    description "hosts level";
                    list host {
                      key cli-id;
                      description "specific host";
                      leaf cli-id {
                        type uint32;
                        mandatory true;
                        description "client id";
```

```
                    }
```

```
                    container duid {
                      description "Sets the DUID";
                      uses duid;
                    }
                    leaf inherit-option-set {
                      type boolean;
                      mandatory true;
                      description "whether to inherit
                        configuration
                        from higher level";
                    }
                    leaf option-set-id {
                      type leafref {
                        path "/server/server-config/option-sets/option-set/
id";
                      }
                      description "The ID field of relevant option-set to be
                        provisioned to clients of this prefix-pool.";
                    }
                    leaf nis-domain-name {
                      type string;
                      description "nis domain name";
                    }
                    leaf nis-plus-domain-name {
                      type string;
                      description "nisp domain name";
                    }
                  }
                }
              }

            }
            container relay-opaque-paras {
              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 relay-name;
                description "relay agents";
                leaf relay-name {
                  type string;
                  mandatory true;
                  description "relay agent name";
                }
                list interface-info {
```

```
                key if-name;
```

```
            description "interface info";
            leaf if-name {
              type string;
              mandatory true;
              description "interface name";
            }
            leaf interface-id {
              type string;
              mandatory true;
              description "interface id";
            }
          }
          list subscribers {
            key subscriber;
            description "subscribers";
            leaf subscriber {
              type uint32;
              mandatory true;
              description "subscriber";
            }
            leaf subscriber-id {
              type string;
              mandatory true;
              description "subscriber id";
            }
          }
          list remote-host {
            key ent-num;
            description "remote host";
            leaf ent-num {
              type uint32;
              mandatory true;
              description "enterprise number";
            }
            leaf remote-id {
              type string;
              mandatory true;
              description "remote id";
            }
          }
        }
      }
      container rsoo-enabled-options {
        description "rsoo enabled options";
        list rsoo-enabled-option{
          key option-code;
          description "rsoo enabled option";
          leaf option-code {
```

```
                type uint16;
                mandatory true;
                description "option code";
              }
              leaf description {
                type string;
                mandatory true;
                description "description of the option";
              }
            }
          }

    }

    /*
     * State data
     */
    container server-state{
      config "false";
      description "states of server";
      container network-ranges{
        description "This model supports a hierarchy
              to achieve dynamic configuration. That is to
              say we could configure the server at different
              levels through this model. The top level is a
              global level which is defined as the container
              'network-ranges'. The following levels are
              defined as sub-containers under it. The
              'network-ranges' contains the parameters
              (e.g. option-sets) that would be allocated to
              all the clients served by this server.";
        list network-range{
          key network-range-id;
          description "The ID field of relevant option-set to be
                  provisioned to clients of this network-range.";
          leaf network-range-id {
            type uint32;
            mandatory true;
            description "equivalent to subnet id";
          }
          container address-pools{
            description "A container that describes
              the DHCPv6 server's address pools";
              list address-pool{
                key pool-id;
                    leaf pool-id {
                        type uint32;
                        mandatory true;
```

```
                description "pool id";
              }
          description "...";
          leaf total-ipv6-count {
            type uint64;
            mandatory true;
            description "how many ipv6 addresses
              are in the pool";
          }
          leaf used-ipv6-count {
            type uint64;
            mandatory true;
            description "how many are allocated";
          }
          leaf address-utilization-ratio {
            type uint16;
            mandatory true;
            description "current address pool utilization ratio";
          }
        }
        list binding-info {
          key cli-id;
          description "A list that records a binding
            information for each DHCPv6 client
            that has already been allocated
            IPv6 addresses.";
          leaf cli-id {
            type uint32;
            mandatory true;
            description "client id";
            }
          container duid {
            description "Read the DUID";
            uses duid;
            }
          list cli-ia{
            key iaid;
            description "client IA";
            leaf ia-type {
              type string;
              mandatory true;
              description "IA type";
            }
            leaf iaid {
              type uint32;
              mandatory true;
              description "IAID";
            }
```

```
        leaf-list cli-addr {
          type inet:ipv6-address;
          description "client addr";
        }
        leaf pool-id {
          type uint32;
          mandatory true;
          description "pool id";
        }
      }
    }
  }
  container prefix-pools{
    description "If a server supports prefix
      delegation function, this container will
      be used to define the delegating router's
      prefix pools.";
    list prefix-pool {
      key pool-id;
      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'.";
      leaf pool-id {
        type uint32;
        mandatory true;
        description "pool id";
        }
      leaf prefix-utilization-ratio {
        type uint16;
        mandatory true;
        description "current prefix pool utilization ratio";
        }
    }
    list binding-info {
      key cli-id;
      description "A list records a binding information
      for each DHCPv6 client that has already been
      alloated IPv6 addresses.";
      leaf cli-id {
        type uint32;
        mandatory true;
        description "client id";
      }
      container duid {
        description "Reads the DUID";
        uses duid;
      }
```

```
                list cli-iapd {
                  key iaid;
                  description "client IAPD";
                  leaf iaid {
                    type uint32;
                    mandatory true;
                    description "IAID";
                  }
                  leaf-list cli-prefix {
                    type inet:ipv6-prefix;
                    description "client ipv6 prefix";
                  }
                  leaf-list cli-prefix-len {
                    type uint8;
                    description "client prefix length";
                  }
                  leaf pool-id {
                    type uint32;
                    mandatory true;
                    description "pool id";
                  }

                }
              }
            }

            list address-prefix-assign-param {
              // Zihao - This probably needs further updated.
              // But is it a way to represent the address/prefix assignment
              // logic?
              key cli-id;
              description "This list includes some parameters/identifiers
                that the server obtains from DHCPv6 options in this network-
range.

                These identifiers may be helpful for the server to assign
                addresses/prefixes.";
              reference "Section 3.12 of RFC7824";
              leaf cli-id {
                type uint32;
                mandatory true;
                description "client id";
              }
              leaf source-ipv6-addr {
                type inet:ipv6-address;
                description "The adrress of the link to which the client
                  is attached.";
              }
              container duid {
```

```
                description "The DUID supplied by the client.";
```

```
                uses duid;
              }
              leaf-list iaid {
                type uint32;
                description "IAID";
              }
              leaf-list preferred-addr {
                type inet:ipv6-address;
                description "The IPv6 address preferred by the client.";
              }
              leaf-list preferred-prefix-len {
                type uint8;
                description "The prefix length preferred by the client.";
              }
              leaf client-fqdn {
                type string;
                description "Fully Qualified Domain Name supplied by the
client.";
              }
              leaf client-link-layer-addr {
                type uint16;
                description "Link-layer address supplied by the client.";
              }
              leaf client-enterprise-number {
                type uint32;
                description "Enterprise number supplied by the client.";
              }
              leaf-list client-sys-archi-type {
                type uint16;
                description "Supported system architecture type supplied by
                  the client";
              }
            }

                }

        }

        container packet-stats {
              description "A container presents
              the packet statistics related to
              the DHCPv6 server.";
            leaf solicit-count {
              type uint32;
              mandatory true;
              description "solicit counter";
            }
            leaf request-count {
```

```
            type uint32;
```

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

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


  /*
   * Notifications
   */

  notification notifications {
    description "dhcpv6 server notification module";
    container dhcpv6-server-event {
      description "dhcpv6 server event";
      container pool-running-out {
        description "raised when the address/prefix pool is going to
          run out.  A threshold for utilization ratio of the pool has
          been defined in the server feature so that it will notify the
          administrator when the utilization ratio reaches the
          threshold, and such threshold is a settable parameter";
      leaf max-address-utilization-ratio {
        type uint16;
        mandatory true;
        description "address pool utilization ratio threshold";
      }
      leaf address-utilization-ratio {
        type uint16;
        mandatory true;
        description "current address pool utilization ratio";
      }
      leaf max-prefix-utilization-ratio {
              type uint16;
              mandatory true;
              description "prefix pool utilization ratio threshold";
            }
      leaf prefix-utilization-ratio {
        type uint16;
        mandatory true;
```

```
        description "current prefix pool utilization ratio";
      }
      container duid {
        description "Sets the DUID";
        uses duid;
      }
      leaf serv-name {
        type string;
        description "server name";
      }
      leaf pool-name {
        type string;
        mandatory true;
        description "pool name";
      }
    }
    container invalid-client-detected {
      description "raised when the server has found a client which
        can be regarded as a potential attacker. Some description
        could also be included.";
      container duid {
        description "Sets the DUID";
        uses duid;
      }
      leaf description {
        type string;
        description "description of the event";
      }
    }
  }
  }
}

<CODE ENDS>
```

3.2. DHCPv6 Relay YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-relay@2017-12-22.yang"
module ietf-dhcpv6-relay {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay";
  prefix "dhcpv6-client";

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

```
  }

  import ietf-dhcpv6-options {
    prefix dhcpv6-options;
    revision-date "2017-12-22";
  }

  organization "DHC WG";
  contact "yong@csnet1.cs.tsinghua.edu.cn
    lh.sunlinh@gmail.com
    ian.farrer@telekom.de
    sladjana.zechlin@telekom.de
    hezihao9512@gmail.com";

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

  revision 2017-12-22 {
    description "Resolve most issues on Ian's github.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  revision 2017-11-24 {
    description "First version of the separated relay specific
      YANG model.";

    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  /*
   * Grouping
   */

  grouping vendor-infor {
   description "Vendor information.";
   container vendor-info {
     description "";
     leaf ent-num {
       type uint32;
       mandatory true;
       description "enterprise number";
     }
     leaf-list data {
       type string;
       description "specific vendor info";
     }
   }
  }
```

```
  /*
   * Data Nodes
   */

container relay {
    presence "Enables the relay";
    description "dhcpv6 relay portion";
    container relay-config{
      description "configuration tree of relay";
        container relay-attributes {
            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;
              description "relay agent name";
            }
            leaf description {
              type string;
              description "description of the relay agent";
            }
            leaf-list dest-addrs {
              type inet:ipv6-address;
              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.";
            }
            list subscribers {
              key subscriber;
              description "subscribers";
              leaf subscriber {
                type uint8;
                mandatory true;
                description "subscriber";
              }
              leaf subscriber-id {
                type string;
                mandatory true;
                description "subscriber id";
              }
            }
```

```
            list remote-host {
              key ent-num;
              description "remote host";
              leaf ent-num {
                type uint32;
                mandatory true;
                description "enterprise number";
              }
              leaf remote-id {
                type string;
                mandatory true;
                description "remote id";
              }
            }
            uses vendor-infor;
          }

          container rsoo-option-sets {
          description "DHCPv6 relay agent could provide
                some information that would be useful to DHCPv6 client.
            Since relay agent cannot provide options directly to
            the client, RSOO-enabled options are defined to
            propose options for the server to send to the client.
            This container models such RSOO-enabled options.";
          reference "RFC6422";
            list option-set {
              key id;
              description "This list under the 'rsoo-option-sets' container
              is similar to the that defined in server module. It allows
              the relay to implement several sets of RSOO-enabled options
              for different interfaces. The list only include the EAP
              Re-authentication Protocol (ERP) Local Domain Name DHCPv6 Option
              defined in RFC6440, since it is the only one
              RSOO-enabled options accepted by IANA so far.";
                leaf id {
                  type uint32;
                  description "option sed id";
                }
              uses dhcpv6-options:relay-supplied-option-definitions;
            }
          }

          list relay-if {
            // if - This should reference an entry in ietf-interfaces
            key if-name;
            description "A relay agent may have several
              interfaces, we should provide a way to configure
              and manage parameters on the interface-level. A
```

```
        list that describes specific interfaces and
        their corresponding parameters is employed to
        fulfil the configfuration. Here we use a string
        called 'if-name; as the key of list.";
      leaf if-name {
        type string;
        mandatory true;
        description "interface name";
      }
      leaf enable {
        type boolean;
        mandatory true;
        description
          "whether this interface is enabled";
      }
      leaf ipv6-address {
        type inet:ipv6-address;
        description
          "ipv6 address for this interface";
      }
      leaf interface-id {
        type string;
        description "interface id";
      }
      leaf rsoo-option-set-id {
        type leafref {
          path "/relay/relay-config/rsoo-option-sets/option-set/id";
        }
        description "Configured Relay Supplied Option set";
      }
      list next-entity {
        key dest-addr;
        description "This node defines
          a list that is used to describe
          the next hop entity of this
          relay distinguished by their
          addresses.";
        leaf dest-addr {
          type inet:ipv6-address;
          mandatory true;
          description "destination addr";
        }
        leaf available {
          type boolean;
          mandatory true;
          description "whether the next entity
            is available or not";
        }
```

```
              leaf multicast {
                type boolean;
                mandatory true;
                description "whether the address is
                  multicast or not";
              }
              leaf server {
                type boolean;
                mandatory true;
                description "whether the next entity
                  is a server";
              }
            }
          }


    }


  container relay-state{
    config "false";
    description "state tree of relay";
    list relay-if{
      key if-name;
      description "...";
      leaf if-name{
        type string;
        mandatory true;
        description "interface name";
      }
            list pd-route {
                // if - need to look at if/how we model these. If they are
                // going to be modelled, then they should be ro state
                // entries (we're not trying to configure routes here)
                key pd-route-id;
                description "pd route";
                leaf pd-route-id {
                  type uint8;
                  mandatory true;
                  description "pd route id";
                }
                leaf requesting-router-id {
                  type uint32;
                  mandatory true;
                  description "requesting router id";
                }
                leaf delegating-router-id {
                  type uint32;
```

```
                mandatory true;
                description "delegating router id";
              }
              leaf next-router {
                type inet:ipv6-address;
                mandatory true;
                description "next router";
              }
              leaf last-router {
                type inet:ipv6-address;
                mandatory true;
                description "previous router";
              }
            }
      list next-entity{
        key dest-addr;
        description "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.";
        leaf dest-addr{
          type inet:ipv6-address;
          mandatory true;
          description "destination addr";
        }
        container packet-stats {
          description "packet statistics";
          leaf solicit-rvd-count {
                      type uint32;
                      mandatory true;
                      description
                        "solicit received counter";
                    }
                  leaf request-rvd-count {
                    type uint32;
                    mandatory true;
                    description
                      "request received counter";
                  }
                  leaf renew-rvd-count {
                    type uint32;
                    mandatory true;
                    description
                      "renew received counter";
                  }
                  leaf rebind-rvd-count {
                    type uint32;
                    mandatory true;
```

```
        description
          "rebind recevied counter";
      }
      leaf decline-rvd-count {
        type uint32;
        mandatory true;
        description
          "decline received counter";
      }
      leaf release-rvd-count {
        type uint32;
        mandatory true;
        description
          "release received counter";
      }
      leaf info-req-rvd-count {
        type uint32;
        mandatory true;
        description
          "information request counter";
      }
      leaf relay-for-rvd-count {
        type uint32;
        mandatory true;
        description
          "relay forward received counter";
      }
      leaf relay-rep-rvd-count {
        type uint32;
        mandatory true;
        description
          "relay reply received counter";
      }
      leaf packet-to-cli-count {
        type uint32;
        mandatory true;
        description
          "packet to client counter";
      }
      leaf adver-sent-count {
        type uint32;
        mandatory true;
        description
          "advertisement sent counter";
      }
      leaf confirm-sent-count {
        type uint32;
        mandatory true;
```

```
                  description
                    "confirm sent counter";
                }
                leaf reply-sent-count {
                  type uint32;
                  mandatory true;
                  description
                    "reply sent counter";
                }
                leaf reconfig-sent-count {
                  type uint32;
                  mandatory true;
                  description
                    "reconfigure sent counter";
                }
                leaf relay-for-sent-count {
                  type uint32;
                  mandatory true;
                  description
                    "relay forward sent counter";
                }
                leaf relay-rep-sent-count {
                  type uint32;
                  mandatory true;
                  description
                    "relay reply sent counter";
                }
        }

      }
    }


      container relay-stats {
          config "false";
          description "relay statistics";
          leaf cli-packet-rvd-count {
            type uint32;
            mandatory true;
            description "client packet received counter";
          }
          leaf relay-for-rvd-count {
            type uint32;
            mandatory true;
            description "relay forward received counter";
          }
          leaf relay-rep-rvd-count {
            type uint32;
```

```
            mandatory true;
            description "relay reply recevied counter";
          }
          leaf packet-to-cli-count {
            type uint32;
            mandatory true;
            description "packet to client counter";
          }
          leaf relay-for-sent-count {
            type uint32;
            mandatory true;
            description "relay forward sent counter";
          }
          leaf relay-rep-sent-count {
            type uint32;
            mandatory true;
            description "relay reply sent counter";
          }
          leaf discarded-packet-count {
            type uint32;
            mandatory true;
            description "discarded packet counter";
          }
        }

  }


  }

  /*
   * Notifications
   */

  notification notifications {
    description "dhcpv6 relay notification module";
    container dhcpv6-relay-event {
      description "dhcpv6 relay event";
      container topo-changed {
        description "raised when the topology
          of the relay agent is changed.";
        leaf relay-if-name {
          type string;
          mandatory true;
          description "relay interface name";
        }
        leaf first-hop {
          type boolean;
```

```
          mandatory true;
          description "first hop";
        }
        leaf last-entity-addr {
          type inet:ipv6-address;
          mandatory true;
          description "last entity address";
        }
      }
    }
  }
}

<CODE ENDS>
```

3.3. DHCPv6 Client YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-client@2017-12-22.yang"
module ietf-dhcpv6-client {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client";
  prefix "dhcpv6-client";

  import ietf-inet-types {
    prefix inet;
    revision-date "2013-07-15";
  }
  import ietf-yang-types {
    prefix yang;
    revision-date "2013-07-15";
  }
  import ietf-dhcpv6-options {
    prefix dhcpv6-options;
    revision-date "2017-12-22";
  }

  organization "DHC WG";
  contact "yong@csnet1.cs.tsinghua.edu.cn
    wangh13@mails.tsinghua.edu.cn
    lh.sunlinh@gmail.com
    ian.farrer@telekom.de
    sladjana.zechlin@telekom.de
    hezihao9512@gmail.com  ";

  description "This model defines a YANG data model that can be
```

```
    used to configure and manage a DHCPv6 client.";

  revision 2017-12-22 {
    description "Resolve most issues on Ian's github.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  revision 2017-11-24 {
      description "First version of the separated client specific
        YANG model.";
      reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  /*
   * Grouping
   */

  grouping vendor-infor {
   description "Vendor information.";
   container vendor-info {
     description "";
     leaf ent-num {
       type uint32;
       mandatory true;
       description "enterprise number";
     }
     leaf-list data {
       type string;
       description "specific vendor info";
     }
   }
  }

  grouping duid {
      description "DHCP Unique Identifier";
      reference "RFC3315: Section 9";
      leaf type-code {
          type uint16;
          default 65535;
          description "Type code of this DUID";
        }
      choice duid-type {
      default duid-invalid;
        description "Selects the format for the DUID.";
        case duid-llt {
          description "DUID Based on Link-layer Address Plus Time (Type 1 -
DUID-LLT)";
          reference "RFC3315 Section 9.2";
```

```
          leaf duid-llt-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
          leaf duid-llt-time {
            type yang:timeticks;
            description "The time value is the time that the DUID is generated
              represented in seconds since midnight (UTC), January 1, 2000,
                          modulo 2^32.";
          }
          leaf duid-llt-link-layer-addr {
            type yang:mac-address;
            description "Link-layer address as described in RFC2464";
          }
        }
        case duid-en {
          description "DUID Assigned by Vendor Based on Enterprise Number (Type
2 - DUID-EN)";
          reference "RFC3315 Section 9.3";
          leaf duid-en-enterprise-number {
            type uint32;
            description "Vendor's registered Private Enterprise Number as
              maintained by IANA";
          }
          leaf duid-en-identifier {
            type string;
            description "Indentifier, unique to the device that is using it";
          }
        }
        case duid-ll {
          description "DUID Based on Link-layer Address (Type 3 - DUID-LL)";
          reference "RFC3315 Section 9.4";
          leaf duid-ll-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
          leaf duid-ll-link-layer-addr {
              type yang:mac-address;
              description "Link-layer address as described in RFC2464";
        }
        }
        case duid-uuid {
          description "DUID Based on Universally Unique Identifier (Type 4 -
DUID-UUID)";
          reference "RFC6335 Defination of the UUID-Based Unique Identifier";
          leaf uuid {
            type yang:uuid;
            description "A Universally Unique IDentifier in the string
```

```
representation
                defined in RFC 4122.  The canonical representation uses
                lowercase characters";
```

```
          }
        }
        case duid-invalid {
          description "DUID based on free raw bytes";
          leaf data {
            type binary;
            description "The bits to be used as the identifier";
          }
        }
      }
    }
  grouping portset-para {
    description "portset parameters";
    container port-parameter {
      description "port parameter";
      leaf offset {
        type uint8;
        mandatory true;
        description "offset in a port set";
      }
      leaf psid-len {
        type uint8;
        mandatory true;
        description "length of a psid";
      }
      leaf psid {
        type uint16;
        mandatory true;
        description "psid value";
      }
    }
  }

  grouping iaid {
    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 {
        config "false";
        key iaid;
        description "IA";
        leaf iaid {
          type uint32;
          mandatory true;
          description "IAID";
        }
        leaf ia-type {
```

```
          type string;
          mandatory true;
          description "IA type";
        }
        leaf-list ipv6-addr {
          type inet:ipv6-address;
          description "ipv6 address";
        }
        leaf-list ipv6-prefix {
          type inet:ipv6-prefix;
          description "ipv6 prefix";
        }
        leaf-list prefix-length {
          type uint8;
          description "ipv6 prefix length";
        }
        leaf t1-time {
          type yang:timeticks;
          mandatory true;
          description "t1 time";
        }
        leaf t2-time {
          type yang:timeticks;
          mandatory true;
          description "t2 time";
        }
        leaf preferred-lifetime {
          type yang:timeticks;
          mandatory true;
          description "preferred lifetime";
        }
        leaf valid-lifetime {
          type yang:timeticks;
          mandatory true;
          description "valid lifetime";
        }
      }

  }

  /*
   * Data Nodes
   */

 container client {
    presence "Enables the client";
    description "dhcpv6 client portion";
    container client-config{
```

```
      description "configuration tree of client";
        container duid {
            description "Sets the DUID";
            uses duid;
          }
          list client-if {
            key if-name;
            description "A client may have several
              interfaces, it is more reasonable to
              configure and manage parameters on
              the interface-level. The list defines
              specific client interfaces and their
              data. Different interfaces are distinguished
              by the key which is a configurable string
              value.";
            leaf if-name {
              type string;
              mandatory true;
              description "interface name";
            }
            leaf cli-id {
              type uint32;
              mandatory true;
              description "client id";
            }
            leaf description {
              type string;
              description
                "description of the client interface";
            }
            leaf pd-function {
              type boolean;
              mandatory true;
              description "Whether the client
                can act as a requesting router
                to request prefixes using prefix
                delegation ([RFC3633]).";
            }
            leaf rapid-commit {
              type boolean;
              mandatory true;
              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.";
            }
            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.";
              // if - not sure about the intended use here as it seems
              // to be redfining what will be received in the PIO. Is
              // the intention to be whether they PIO options will be
              // obeyed as received or overridden?
            leaf m-tab {
              type boolean;
              mandatory true;
              description "m tab";
            }
            leaf o-tab {
              type boolean;
              mandatory true;
              description "o tab";
            }
          }
          container client-configured-options {
            description "client configured options";
            uses dhcpv6-options:client-option-definitions;
          }
    }


    }

    container client-state{
      config "false";
      description "state tree of client";
        container if-other-paras {
            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.";
            uses dhcpv6-options:server-option-definitions;
          }
    }
  }
```

```
  /*
   * Notifications
   */

  notification notifications {
  description "dhcpv6 client notification module";
    container dhcpv6-client-event {
      description "dhcpv6 client event";
      container ia-lease-event {
        description "raised when the
          client was allocated a new IA from
          the server or it renew/rebind/release
          its current IA";
        leaf event-type {
          type enumeration{
            enum "allocation" {
              description "allocate";
            }
            enum "rebind" {
              description "rebind";
            }
            enum "renew" {
              description "renew";
            }
            enum "release" {
              description "release";
            }
          }
          mandatory true;
          description "event type";
        }
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf iaid {
          type uint32;
          mandatory true;
          description "IAID";
        }
        leaf serv-name {
          type string;
          description "server name";
        }
        leaf description {
          type string;
          description "description of event";
        }
```

```
      }
      container invalid-ia-detected {
        description "raised when the identity
          association of the client can be proved
          to be invalid.  Possible condition includes
          duplicated address, illegal address, etc.";
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf cli-duid {
          type uint32;
          mandatory true;
          description "duid of client";
        }
        leaf iaid {
          type uint32;
          mandatory true;
          description "IAID";
        }
        leaf serv-name {
          type string;
          description "server name";
        }
        leaf description {
          type string;
          description "description of the event";
        }
      }
      container retransmission-failed {
        description "raised when the retransmission
          mechanism defined in [RFC3315] is failed.";
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf description {
          type enumeration {
            enum "MRC failed" {
              description "MRC failed";
            }
            enum "MRD failed" {
              description "MRD failed";
            }
          }
          mandatory true;
          description "description of failure";
        }
```

```
      }
      container failed-status-turn-up {
        description "raised when the client receives
          a message includes an unsuccessful Status Code
          option.";
        container duid {
          description "Sets the DUID";
          uses duid;
        }
        leaf status-code {
          type enumeration {
            enum "1" {
              description "UnspecFail";
            }
            enum "2" {
              description "NoAddrAvail";
            }
            enum "3" {
              description "NoBinding";
            }
            enum "4" {
              description "NotOnLink";
            }
            enum "5" {
              description "UseMulticast";
            }
          }
          mandatory true;
          description "employed status code";
        }
      }
    }
  }
}

<CODE ENDS>
```

3.4. DHCPv6 Options YANG Model

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-options@2017-12-22.yang"
module ietf-dhcpv6-options {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-options";
  prefix "dhcpv6-options";

  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
    ian.farrer@telekom.de
    sladjana.zechlin@telekom.de
    hezihao9512@gmail.com";

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

  revision 2017-12-22 {
    description "Resolve most issues on Ian's github.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  revision 2017-11-24 {
    description "First version of the separated DHCPv6 options
      YANG model.";

    reference "I-D:draft-ietf-dhc-dhcpv6-yang";
  }

  /*
   * Features
   */


  //  features for server options
  feature server-unicast-op {
    description "Support for Server Unicast option";
  }
  feature sip-server-domain-name-list-op {
    description "Support for SIP Server Domain Name List option";
  }
  feature sip-server-address-list-op {
    description "Support for SIP Server Address List option";
  }
  feature dns-config-op {
    description "Support for DNS Recursive Name Server option";
```

```
  }
  feature domain-searchlist-op {
    description "Support for Domain Search List Option";
  }
  feature nis-config-op {
    description "Support for Network Information Service (NIS)
      Servers option";
  }
  feature nis-plus-config-op {
    description "Support for Network Information Service V2 (NIS+)
      Servers option";
  }
  feature nis-domain-name-op {
    description "Support for Network Information Service (NIS)
      Domain Name option";
  }
  feature nis-plus-domain-name-op {
    description "Support for Network Information Service V2 (NIS+)
      Server option";
  }
  feature sntp-server-op {
    description "Support for Simple Network Protocol Configuration
      (SNTP) Servers option";
  }
  feature info-refresh-time-op {
    description "Support for Information Refresh Time option";
  }
  feature client-fqdn-op {
    description "Support for Client FQDN option";
  }
  feature posix-timezone-op {
    description "Support for New POIX Timezone option";
  }
  feature tzdb-timezone-op {
    description "Support for New TZDB Timezone option";
  }
  feature ntp-server-op {
    description "Support for Network Time Protocol (NTP)
      Server option";
  }
  feature boot-file-url-op {
    description "Support for Boot File URL option";
  }
  feature boot-file-param-op {
    description "Support for Boot File Parameters option";
  }
  feature aftr-name-op {
    description "Support for Address Family Transition
```

```
      Router (AFTR) option";
  }
  feature kbr-default-name-op {
    description "Support for Kerberos Default Name
      Option";
  }
  feature kbr-kdc-op {
    description "Support for Kerberos KDC option";
  }
  feature sol-max-rt-op {
    description "Support for SOL_MAX_RT option";
  }
  feature inf-max-rt-op {
    description "Support for INF_MAX_RT option";
  }
  feature addr-selection-op {
    description "Support for Address Selection opiton";
  }
  feature pcp-server-op {
    description "Support for Port Control Protocol (PCP)
      option";
  }
  feature s46-rule-op {
    description "Support for S46 Rule option";
  }
  feature s46-br-op {
    description "Support for S46 Border Relay (BR) option";
  }
  feature s46-dmr-op {
    description "Support for S46 Default Mapping Rule
      (DMR) option";
  }
  feature s46-v4-v6-binding-op {
    description "Support for S46 IPv4/IPv6 Address
      Bind option";
  }

  // features for relay-supplied options
  feature erp-local-domain-name-op {
    description "Support for ERP Local Domain
      Name option";
  }

  // features for client options
  feature option-request-op {
    description "Support for Option Request option";
  }
  feature rapid-commit-op {
```

```
    description "Support for Rapid Commit option";
  }
  feature user-class-op {
    description "Support for User Class option";
  }
  feature vendor-class-op {
    description "Support for Vendor Class option";
  }
  feature client-arch-type-op {
    description "Support for Client System Architecture
      Type option";
  }
  feature client-network-interface-identifier-op {
    description "Support for Client Network Interface
      Identifier option";
  }
  feature kbr-principal-name-op {
    description "Support for Kerberos Principal
      Name option";
  }
  feature kbr-realm-name-op {
    description "Support Kerberos Realm Name option";
  }
  feature client-link-layer-addr-op {
    description "Support for Client Link-Layer Address
      Option";
  }

  // features for custom options
  feature operator-op-ipv6-address {
    description "Support for Option with IPv6 Addresses";
  }
  feature operator-op-single-flag {
    description "Support for Option with Single Flag";
  }
  feature operator-op-ipv6-prefix {
    description "Support for Option with IPv6 Prefix";
  }
  feature operator-op-int32 {
    description "Support for Opion with 32-bit
      Integer Value";
  }
  feature operator-op-int16 {
    description "Support for Opion with 16-bit
      Integer Value";
  }
  feature operator-op-int8 {
    description "Support for Opion with 8-bit
```

```
      Integer Value";
  }
  feature operator-op-uri {
    description "Support for Opion with URI";
  }
  feature operator-op-textstring {
    description "Support for Opion with Text
      String";
  }
  feature operator-op-var-data {
    description "Support for Opion with
      Variable-Length Data";
  }
  feature operator-op-dns-wire {
    description "Support for Opion with DNS Wire
      Format Domain Name List";
  }

  /*
   * Groupings
   */

  grouping portset-para {
   description "portset parameters";
   container port-parameter {
     description "port parameter";
     leaf offset {
       type uint8;
       mandatory true;
       description "offset in a port set";
     }
     leaf psid-len {
       type uint8;
       mandatory true;
       description "length of a psid";
     }
     leaf psid {
       type uint16;
       mandatory true;
       description "psid value";
     }
   }
  }

  grouping duid {
      description "DHCP Unique Identifier";
      reference "RFC3315: Section 9";
      leaf type-code {
```

```
          type uint16;
          default 65535;
          description "Type code of this DUID";
        }
      choice duid-type {
      default duid-invalid;
        description "Selects the format for the DUID.";
        case duid-llt {
          description "DUID Based on Link-layer Address Plus Time (Type 1 -
DUID-LLT)";
          reference "RFC3315 Section 9.2";

          leaf duid-llt-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
          leaf duid-llt-time {
            type yang:timeticks;
            description "The time value is the time that the DUID is generated
              represented in seconds since midnight (UTC), January 1, 2000,
                          modulo 2^32.";
          }
          leaf duid-llt-link-layer-addr {
            type yang:mac-address;
            description "Link-layer address as described in RFC2464";
          }
        }
        case duid-en {
          description "DUID Assigned by Vendor Based on Enterprise Number (Type
2 - DUID-EN)";
          reference "RFC3315 Section 9.3";
          leaf duid-en-enterprise-number {
            type uint32;
            description "Vendor's registered Private Enterprise Number as
              maintained by IANA";
          }
          leaf duid-en-identifier {
            type string;
            description "Indentifier, unique to the device that is using it";
          }
        }
        case duid-ll {
          description "DUID Based on Link-layer Address (Type 3 - DUID-LL)";
          reference "RFC3315 Section 9.4";
          leaf duid-ll-hardware-type {
            type uint16;
            description "Hardware type as assigned by IANA (RFC826).";
          }
```

```
          leaf duid-ll-link-layer-addr {
              type yang:mac-address;
```

```
              description "Link-layer address as described in RFC2464";
        }
        }
        case duid-uuid {
          description "DUID Based on Universally Unique Identifier (Type 4 -
DUID-UUID)";
          reference "RFC6335 Defination of the UUID-Based Unique Identifier";
          leaf uuid {
            type yang:uuid;
            description "A Universally Unique IDentifier in the string
representation
                defined in RFC 4122.  The canonical representation uses
                lowercase characters";
          }
        }
        case duid-invalid {
          description "DUID based on free raw bytes";
          leaf data {
            type binary;
            description "The bits to be used as the identifier";
          }
        }
      }
    }

  grouping server-option-definitions {
    description "Contains definitions for options configured on the
      DHCPv6 server which will be supplied to clients.";
    container server-unicast-option {
      if-feature server-unicast-op;
      presence "Enable this option";
      description "OPTION_UNICAST (12) Server Unicast Option";
      reference "RFC3315: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6)";
      leaf server-address {
        type inet:ipv6-address;
        description "server ipv6 address";
      }
    }
    container sip-server-domain-name-list-option {
        if-feature sip-server-domain-name-list-op;
        presence "Enable this option";
        description "OPTION_SIP_SERVER_D (21) SIP Servers Domain Name List";
        reference "RFC3319: Dynamic Host Configuration Protocol
          (DHCPv6) Options for Session Initiation Protocol (SIP)
          Servers";
          leaf sip-serv-domain-name {
            type string;
```

```
            mandatory true;
            description "sip server domain
```

```
            name";
          }
    }
    container sip-server-address-list-option {
        if-feature sip-server-address-list-op;
        presence "Enable this option";
        description "OPTION_SIP_SERVER_A (22) SIP Servers IPv6 Address List";
        reference "RFC3319: Dynamic Host Configuration Protocol
          (DHCPv6) Options for Session Initiation Protocol (SIP)
          Servers";
        list sip-server {
          key sip-serv-id;
          description "sip server info";
          leaf sip-serv-id {
            type uint8;
            mandatory true;
            description "sip server id";
          }
          leaf sip-serv-addr {
            type inet:ipv6-address;
            mandatory true;
            description "sip server addr";
          }
        }
    }
    container dns-config-option {
      if-feature dns-config-op;
      presence "Enable this option";
      description "OPTION_DNS_SERVERS (23) DNS recursive Name
        Server option";
      reference "RFC3646: DNS Configuration options for Dynamic
        Host Configuration Protocol for IPv6 (DHCPv6)";
      list dns-server {
        key dns-serv-id;
        description "dns server info";
        leaf dns-serv-id {
          type uint8;
          mandatory true;
          description "DNS server list entry ID.";
        }
        leaf dns-serv-addr {
          type inet:ipv6-address;
          mandatory true;
          description "DNS server address.";
        }
      }
    }
    container domain-searchlist-option {
```

```
      if-feature domain-searchlist-op;
      presence "Enable this option";
      description "OPTION_DOMAIN_LIST (24) Domain Search List Option";
      reference "RFC3646: DNS Configuration options for Dynamic
        Host Configuration Protocol for IPv6 (DHCPv6)";
      list domain-searchlist {
        key domain-searchlist-id;
        description "dns server info";
        leaf domain-searchlist-id {
          type uint8;
          mandatory true;
          description "Domain seachlist entry ID.";
        }
        leaf domain-search-list-entry {
          type string;
          mandatory true;
          description "Domain search list entry.";
        }
      }
    }
    container nis-config-option {
      if-feature nis-config-op;
      presence "Enable this option";
      description "OPTION_NIS_SERVERS (27) Network Information Service (NIS)
        Servers Option.";
      reference "RFC3898: Network Information Service (NIS) Configuration
        Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)";
      list nis-server {
        key nis-serv-id;
        description "nis server info";
        leaf nis-serv-id {
          type uint8;
          mandatory true;
          description "nis server id";
        }
        leaf nis-serv-addr {
          type inet:ipv6-address;
          mandatory true;
          description "nis server addr";
        }
      }
    }
    container nis-plus-config-option {
      if-feature nis-plus-config-op;
      presence "Enable this option";
      description "OPTION_NISP_SERVERS (28): Network Information Service V2
        (NIS+) Servers Option.";
      reference "RFC3989: Network Information Service (NIS) Configuration
```

```
        Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)";
      list nis-plus-server {
        key nis-plus-serv-id;
        description "NIS+ server information.";
        leaf nis-plus-serv-id {
          type uint8;
          mandatory true;
          description "nisp server id";
        }
        leaf nis-plus-serv-addr {
          type inet:ipv6-address;
          mandatory true;
          description "nisp server addr";
        }
      }
    }
    container nis-domain-name-option {
      if-feature nis-domain-name-op;
      presence "Enable this option";
      description "OPTION_NIS_DOMAIN_NAME (29) Network Information
        Service (NIS) Domain Name Option";
      reference "RFC3989: Network Information Service (NIS)
        Configuration Options for Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6)";
      leaf nis-domain-name {
        type string;
        description "The Network Information Service (NIS) Domain Name
        option is used by the server to convey client's NIS Domain Name
        info to the client.";
      }
    }
    container nis-plus-domain-name-option {
        if-feature nis-plus-domain-name-op;
        presence "Enable this option";
        description "OPTION_NISP_DOMAIN_NAME (30) Network Information
          Service V2 (NIS+) Domain Name Option";
        reference "RFC3989: Network Information Service (NIS)
          Configuration Options for Dynamic Host Configuration Protocol
          for IPv6 (DHCPv6)";
        leaf nis-plus-domain-name {
          type string;
          description "The Network Information Service V2 (NIS+) Domain Name
          option is used by the server to convey client's NIS+ Domain Name
          info to the client.";
        }
      }
    container sntp-server-option {
      if-feature sntp-server-op;
```

```
      presence "Enable this option";
      description "OPTION_SNTP_SERVERS (31) Simple Network Time Protocol
        (SNTP) Servers Option";
      reference "RFC4075: Simple Network Time Protocol (SNTP) Configuration
        Option for DHCPv6";
      list sntp-server {
        key sntp-serv-id;
        description "sntp server info";
        leaf sntp-serv-id {
          type uint8;
          mandatory true;
          description "sntp server id";
        }
        leaf sntp-serv-addr {
          type inet:ipv6-address;
          mandatory true;
          description "sntp server addr";
        }
      }
    }
    container info-refresh-time-option {
      if-feature info-refresh-time-op;
      presence "Enable this option";
      description "OPTION_INFORMATION_REFRESH_TIME (32) Information Refresh
        Time option.";
      reference "RFC4242: Information Refresh Time Option for Dynamic Host
        Configuration Protocol for IPv6 (DHCPv6";
      leaf info-refresh-time {
        type yang:timeticks;
        mandatory true;
        description "The refresh time.";
      }
    }
    container client-fqdn-option {
      if-feature client-fqdn-op;
      presence "Enable this option";
      description "OPTION_CLIENT_FQDN (39) DHCPv6 Client FQDN Option";
      reference "RFC4704: The Dynamic Host Configuration Protocol for IPv6
        (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option";
      leaf server-initiate-update {
        type boolean;
        mandatory true;
        description "server initiate";
      }
      leaf client-initiate-update {
        type boolean;
        mandatory true;
        description "client initiate";
```

```
      }
      leaf modify-name-from-cli {
        type boolean;
        mandatory true;
        description "modify by client";
      }
    }
    container posix-timezone-option {
      if-feature posix-timezone-op;
      presence "Enable this option";
      description "OPTION_NEW_POSIX_TIMEZONE (41) Posix Timezone option";
      reference "RFC4833: Timezone Options for DHCP";
      leaf tz-posix {
        type string;
        mandatory true;
        description "TZ Posix IEEE 1003.1 String";
      }
    }
    container tzdb-timezone-option {
      if-feature tzdb-timezone-op;
      presence "Enable this option";
      description "OPTION_NEW_TZDB_TIMEZONE (42) Timezone Database option";
      reference "RFC4822: Timezone Options for DHCP";
      leaf tz-database {
        type string;
        mandatory true;
        description "Reference to the TZ Database";
      }
    }
    container ntp-server-option {
      //This option looks like it needs work to correctly model the
      //option as defined in the RFC.

      // Zihao - Re-modelled so it only contains one
      // time source suboption

      if-feature ntp-server-op;
      presence "Enable this option";
      description "OPTION_NTP_SERVER (56) NTP Server Option for DHCPv6";
      reference "RFC5908: Network Time Protocol (NTP) Server Option for
      DHCPv6";
      list ntp-server {
        key ntp-serv-id;
        description "ntp server info";
        leaf ntp-serv-id {
          type uint8;
          mandatory true;
          description "ntp server id";
```

```
        }
        choice ntp-time-source-suboption {
          description "Select a NTP time source suboption.";
          case server-address {
        leaf-list ntp-serv-addr-suboption {
                  type inet:ipv6-address;
                  description "ntp server addr";
                }
          }
          case server-multicast-address {
            leaf-list ntp-serv-mul-addr-suboption {
                    type inet:ipv6-address;
                    description "ntp server multicast addr";
                  }
          }
          case server-fqdn {
            leaf-list ntp-serv-fqdn-suboption {
                    type string;
                    description "ntp server fqdn";
                  }
          }
        }
      }
    }
    container boot-file-url-option {
      if-feature boot-file-url-op;
      presence "Enable this option";
      description "OPT_BOOTFILE_URL (59) Boot File URL option";
      reference "RFC5970: DHCPv6 Options for Network Boot";
      list boot-file {
        key boot-file-id;
        description "boot file info";
        leaf boot-file-id {
          type uint8;
          mandatory true;
          description "boot file id";
        }
            leaf-list suitable-arch-type {
                type uint16;
                description "architecture type";
              }
              leaf-list suitable-net-if {
                type uint32;
                description "network interface";
              }
        leaf boot-file-url {
          type string;
          mandatory true;
```

```
          description "url for boot file";
        }
      }
    }
    container boot-file-param-option {
      if-feature boot-file-param-op;
      presence "Enable this option";
      description "OPT_BOOTFiLE_PARAM (60) Boot File Parameters Option";
      reference "RFC5970: DHCPv6 Options for Network Boot";
          list boot-file-paras {
              key para-id;
              description "boot file parameters";
              leaf para-id {
                type uint8;
                mandatory true;
                description "parameter id";
              }
              leaf parameter {
                type string;
                mandatory true;
                description "parameter
                  value";
              }

            }
    }
    container aftr-name-option {
      if-feature aftr-name-op;
      presence "Enable this option";
      description "OPTION_AFTR_NAME (64) AFTR-Name DHCPv6 Option";
      reference "RFC6334: Dynamic Host Configuration Protocol for IPv6
      (DHCPv6) Option for Dual-Stack Lite";
      leaf tunnel-endpoint-name {
        type string;
        mandatory true;
        description "aftr name";
      }
    }
    container kbr-default-name-option {
      if-feature kbr-default-name-op;
      presence "Enable this option";
      description "OPTION_KRB_DEFAULT_REALM_NAME (77) Kerberos Default Realm
Name Option";
      reference "RFC6784: Kerberos Options for DHCPv6";
        leaf default-realm-name {
            type string;
            mandatory true;
            description "default realm name";
```

```
          }
```

```
    }
    container kbr-kdc-option {
      if-feature kbr-kdc-op;
      presence "Enable this option";
      description "OPTION_KRB_KDC (78) Kerberos KDB Option";
      reference "RFC6784: Kerberos Options for DHCPv6";
        list kdc-info {
            key kdc-id;
            description "kdc info";
            leaf kdc-id {
              type uint8;
              mandatory true;
              description "kdc id";
            }
            leaf priority {
              type uint16;
              mandatory true;
              description "priority";
            }
            leaf weight {
              type uint16;
              mandatory true;
              description "weight";
            }
            leaf transport-type {
              type uint8;
              mandatory true;
              description "transport type";
            }
            leaf port-number {
              type uint16;
              mandatory true;
              description "port number";
            }
            leaf kdc-ipv6-addr {
              type inet:ipv6-address;
              mandatory true;
              description "kdc ipv6 addr";
            }
            leaf realm-name {
              type string;
              mandatory true;
              description "realm name";
            }
          }
    }
    container sol-max-rt-option {
      if-feature sol-max-rt-op;
```

```
      presence "Enable this option";
      description "OPTION_SOL_MAX_RT (82) sol max rt option";
      reference "RFC7083: Modification to Default Values of
        SOL_MAX_RT and INF_MAX_RT";
        leaf sol-max-rt-value {
          type yang:timeticks;
          mandatory true;
          description "sol max rt value";
        }
    }
    container inf-max-rt-option {
      if-feature inf-max-rt-op;
      presence "Enable this option";
      description "OPTION_INF_MAX_RT (83) inf max rt option";
      reference "RFC7083: Modification to Default Values of
        SOL_MAX_RT and INF_MAX_RT";
      leaf inf-max-rt-value {
        type yang:timeticks;
        mandatory true;
        description "inf max rt value";
      }
    }
    container addr-selection-option {
      if-feature addr-selection-op;
      presence "Enable this option";
      description "OPTION_ADDRSEL (84) and OPTION_ADDRSEL_TABLE (85)";
      reference "RFC7078: Distributing Address Selection Policy Using
      DHCPv6";
      // if - Needs checking to see if this matches the RFC - there
      // are two options here.
      // Zihao - I think this matches RFC7078
      leaf a-bit-set {
        type boolean;
        mandatory true;
        description "a bit";
      }
      leaf p-bit-set {
        type boolean;
        mandatory true;
        description "p bit";
      }
      list policy-table {
        key policy-id;
        description "policy table";
        leaf policy-id {
          type uint8;
          mandatory true;
          description "policy id";
```

```
        }
        leaf label {
          type uint8;
          mandatory true;
          description "label";
        }
        leaf precedence {
          type uint8;
          mandatory true;
          description "precedence";
        }
        leaf prefix-len {
          type uint8;
          mandatory true;
          description "prefix length";
        }
        leaf prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "prefix";
        }
      }
    }
    container pcp-server-option {
      if-feature pcp-server-op;
      presence "Enable this option";
      description "OPTION_V6_PCP_SERVER (86)
        pcp server option";
      reference "RFC7291: DHCP Options for the Port Control
        Protocol (PCP)";
      list pcp-server {
        key pcp-serv-id;
        description "pcp server info";
        leaf pcp-serv-id {
          type uint8;
          mandatory true;
          description "pcp server id";
        }
        leaf pcp-serv-addr {
          type inet:ipv6-address;
          mandatory true;
          description "pcp server addr";
        }
      }
    }
    container s46-rule-option {
      if-feature s46-rule-op;
      presence "Enable this option";
```

```
      description "OPTION_S46_RULE (89) S46 rule option";
      reference "RFC7598: DHCPv6 Options for Configuration of
        Softwire Address and Port-Mapped Clients";
      list s46-rule {
        key rule-id;
        description "s46 rule";
        leaf rule-id {
          type uint8;
          mandatory true;
          description "rule id";
        }
        leaf rule-type {
          type enumeration {
            enum "BMR" {
            description "BMR";
          }
          enum "FMR" {
            description "FMR";
          }
        }
        mandatory true;
        description "rule type";
      }
      leaf prefix4-len {
        type uint8;
        mandatory true;
        description "ipv4 prefix length";
      }
        leaf ipv4-prefix {
          type inet:ipv4-prefix;
          mandatory true;
          description "ipv4 prefix";
        }
        leaf prefix6-len {
          type uint8;
          mandatory true;
          description "ipv6 prefix length";
        }
        leaf ipv6-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "ipv6 prefix";
        }
        uses portset-para;
      }
    }
    container s46-br-option {
      if-feature s46-br-op;
```

```
      presence "Enable this option";
      description "OPTION_S46_BR (90) S46 BR Option";
      reference "RFC7598: DHCPv6 Options for Configuration of
        Softwire Address and Port-Mapped Clients";
      list br {
        key br-id;
        description "br info";
        leaf br-id {
          type uint8;
          mandatory true;
          description "br id";
        }
        leaf br-ipv6-addr {
          type inet:ipv6-address;
          mandatory true;
          description "br ipv6 addr";
        }
      }
    }
    container s46-dmr-option {
      if-feature s46-dmr-op;
      presence "Enable this option";
      description "OPTION_S46_DMR (91) S46 DMR Option";
      reference "RFC7598: DHCPv6 Options for Configuration of
        Softwire Address and Port-Mapped Clients";
      list dmr {
        key dmr-id;
        description "dmr info";
        leaf dmr-id {
          type uint8;
          mandatory true;
          description "dmr id";
        }
        leaf dmr-prefix-len {
          type uint8;
          mandatory true;
          description "dmr prefix length";
        }
        leaf dmr-ipv6-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "dmr ipv6 prefix";
        }
      }
    }
    container s46-v4-v6-binding-option {
      if-feature s46-v4-v6-binding-op;
      presence "Enable this option";
```

```
      description "OPTION_S46_V4V6BIND (92) S46 IPv4/IPv6 Address
        Binding option";
      reference "RFC7598: DHCPv6 Options for Configuration of
        Softwire Address and Port-Mapped Clients";
      list ce {
        key ce-id;
        description "ce info";
        leaf ce-id {
          type uint8;
          mandatory true;
          description "ce id";
        }
        leaf ipv4-addr {
          type inet:ipv4-address;
          mandatory true;
          description "ce ipv4 addr";
        }
        leaf bind-prefix6-len {
          type uint8;
          mandatory true;
          description "bind ipv6 prefix
            length";
        }
        leaf bind-ipv6-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "bind ipv6 prefix";
        }
        uses portset-para;
      }
    }
  }
  //if - NB - The list of options needs to be updated.

  grouping relay-supplied-option-definitions {
    // if - The structure here needs to be checked and probably reworked.
    description "OPTION_RSOO (66) Relay-Supplied Options option";
    reference "RFC6422: Relay-Supplied DHCP Options";
    container erp-local-domain-name-option {
      if-feature erp-local-domain-name-op;
      presence "Enable this option";
      description "OPTION_ERP_LOCAL_DOMAIN_NAME (65) DHCPv6 ERP Local
        Domain Name Option";
      reference "RFC6440: The EAP Re-authentication Protocol (ERP)
        Local Domain Name DHCPv6 Option";
      list erp-for-client {
        key cli-id;
        description "erp for client";
```

```
        leaf cli-id {
          type uint32;
          mandatory true;
          description "client id";
        }
        container duid {
          description "Sets the DUID";
          // uses duid;
          // if - Maybe DUID definition needs to be moved to this module.
          uses duid;
        }
        leaf erp-name {
          type string;
          mandatory true;
          description "erp name";
        }
      }
    }
  }

  grouping client-option-definitions {
    description "Contains definitions for options configured on the
      DHCPv6 client which will be sent to the server.";
    list new-or-standard-cli-option {
      key option-code;
      description "new or standard client option";
      leaf option-code {
        type uint16;
        mandatory true;
        description "option code";
      }
      leaf option-name {
        type string;
        mandatory true;
        description "option name";
      }
      leaf option-description {
        type string;
        mandatory true;
        description "description of client
          option";
      }
      leaf option-reference {
        type string;
        description "the reference of option";
      }
      leaf option-value {
        type string;
```

```
        mandatory true;
        description "the option value";
      }
    }
    container option-request-option {
      if-feature option-request-op;
      presence "Enable this option";
      description "OPTION_ORO (6) Option Request Option";
      reference "RFC3315: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6)";
      list oro-option {
        key option-code;
        description "oro option";
        leaf option-code {
          type uint16;
          mandatory true;
          description "option code";
        }
        leaf description {
          type string;
          mandatory true;
          description "description of oro
            options";
        }
      }
    }

    container user-class-option {
      if-feature user-class-op;
      presence "Enable this option";
      description "OPTION_USER_CLASS (15) User Class Option";
      reference "RFC3315: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6)";
      list user-class {
        key user-class-id;
        description "user class";
        leaf user-class-id {
          type uint8;
          mandatory true;
          description "user class id";
        }
        leaf user-class-data {
          type string;
          mandatory true;
          description "The information contained in the data area
            of this option is contained in one or more opaque
            fields that represent the user class or classes of
            which the client is a member. ";
```

```
        }
      }
    }
    container vendor-class-option {
      if-feature vendor-class-op;
      presence "Enable this option";
      description "OPTION_VENDOR_CLASS (16) Vendor Class Option";
      reference "RFC3315: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6)";
      leaf enterprise-number {
        type uint32;
        mandatory true;
        description "enterprise number";
      }
      list vendor-class {
        key vendor-class-id;
        description "vendor class";
          leaf vendor-class-id {
              type uint8;
              mandatory true;
              description "vendor class id";
            }
          leaf vendor-class-data {
            type string;
            mandatory true;
            description "The vendor-class-data is composed of a series
                  of separate items, each of which describes some
                  characteristic of the client's hardware configuration.
                  Examples of vendor-class-data instances might include the
                  version of the operating system the client is running or
                  the amount of memory installed on the client.";
          }
      }
    }
    container client-fqdn-option {
      if-feature client-fqdn-op;
      presence "Enable this option";
      description "OPTION_CLIENT_FQDN (39) The Dynamic Host
        Configuration Protocol for IPv6 (DHCPv6) Client Fully
        Qualified Domain Name (FQDN) Option";
      reference "RFC4704: The Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN)
        Option";
      leaf fqdn {
        type string;
        mandatory true;
        description "fqdn";
      }
```

```
      leaf server-initiate-update {
        type boolean;
        mandatory true;
        description "whether server initiate";
      }
      leaf client-initiate-update {
        type boolean;
        mandatory true;
        description "whether client initiate";
      }
    }
    container client-arch-type-option {
      if-feature client-arch-type-op;
      presence "Enable this option";
      description "OPTION_CLIENT_ARCH_TYPE (61) Client System
        Architecture Type Option";
      reference "RFC5970: DHCPv6 Options for Network Boot";
      list architecture-types {
        key type-id;
        description "architecture types";
        leaf type-id {
          type uint16;
          mandatory true;
          description "type id";
        }
        leaf most-preferred {
          type boolean;
          mandatory true;
          description "most preferred flag";
        }
      }
    }
    container client-network-interface-identifier-option {
      if-feature client-network-interface-identifier-op;
      presence "Enable this option";
      description "OPTION_NII (62) Client Network Interface
        Identifier Option";
      reference "RFC5970: DHCPv6 Options for Network Boot";
      leaf type {
        type uint8;
        mandatory true;
        description "type";
      }
      leaf major {
        type uint8;
        mandatory true;
        description "major";
      }
```

```
      leaf minor {
        type uint8;
        mandatory true;
        description "minor";
      }
    }
    container kbr-principal-name-option {
      if-feature kbr-principal-name-op;
      presence "Enable this option";
      description "OPTION_KRB_PRINCIPAL_NAME (75) Kerberos
        Principal Name Option";
      reference "RFC6784: Kerberos Options for DHCPv6";
      list principle-name {
        key principle-name-id;
        description "principle name";
        leaf principle-name-id {
          type uint8;
          mandatory true;
          description "principle name id";
        }
        leaf name-type {
          type int32;
          mandatory true;
          description "This field specifies the type of name that follows.";
        }
        leaf name-string {
          type string;
          mandatory true;
          description "This field encodes a sequence of components that form
            a name, each component encoded as a KerberoString";
        }
      }
      }
    container kbr-realm-name-option {
      if-feature kbr-realm-name-op;
      presence "Enable this option";
      description "OPTION_KRB_REALM_NAME (76) Kerberos Realm Name Option";
      reference "RFC6784: Kerberos Options for DHCPv6";
      leaf realm-name {
        type string;
        mandatory true;
        description "realm name";
      }
    }
    container client-link-layer-addr-option {
      if-feature client-link-layer-addr-op;
      presence "Enable this option";
      description "OPTION_CLIENT_LINKLAYER_ADDR (79) DHCPv6 Client
```

```
        Link-Layer Address Option";
      reference "RFC6939: Client Link-Layer Address Option in
        DHCPv6";
      leaf link-layer-type {
        type uint16;
        mandatory true;
        description "Client link-layer address type. The link-layer
          type MUST be a valid hardware type assigned by the IANA,
          as described in [RFC0826]";
      }
      leaf link-layer-addr {
        type string;
        mandatory true;
        description "Client link-layer address";
      }
    }
    }


  grouping custom-option-definitions {
  description "operator customized options";
    container operator-option-ipv6-address {
      if-feature operator-op-ipv6-address;
      presence "Enable this option";
      description "operator ipv6 address option";
      reference "RFC7227: Guidelines for Creating New DHCPv6 Options";
      list operator-ipv6-addr {
        key operator-ipv6-addr-id;
        description "operator ipv6 address info";
        leaf operator-ipv6-addr-id {
          type uint8;
          mandatory true;
          description "operator ipv6 address id";
        }
        leaf operator-ipv6-addr {
          type inet:ipv6-address;
          mandatory true;
          description "operator ipv6 address id";
        }
      }
    }
    container operator-option-single-flag {
      if-feature operator-op-single-flag;
      presence "Enable this option";
      description "operator single flag";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
```

```
      list flag {
        key flag-id;
        description "operator single flag info";
        leaf flag-id {
          type uint8;
          mandatory true;
          description "operator single flag id";
        }
        leaf flag-value{
          type boolean;
          mandatory true;
          description "operator single flag value";
        }
      }
    }
    container operator-option-ipv6-prefix {
      if-feature operator-op-ipv6-prefix;
      presence "Enable this option";
      description "operator ipv6 prefix option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list operator-ipv6-prefix{
        key operator-ipv6-prefix-id;
        description "operator ipv6 prefix info";
        leaf operator-ipv6-prefix-id {
          type uint8;
          mandatory true;
          description "operator ipv6 prefix id";
        }
        leaf operator-ipv6-prefix6-len{
          type uint8;
          mandatory true;
          description "operator ipv6 prefix length";
        }
        leaf operator-ipv6-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description "operator ipv6 prefix";
        }
      }
    }
    container operator-option-int32 {
      if-feature operator-op-int32;
      presence "Enable this option";
      description "operator integer 32 option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list int32val{
```

```
        key int32val-id;
        description "operator integer 32 info";
        leaf int32val-id {
          type uint8;
          mandatory true;
          description "operator integer 32 id";
        }
        leaf int32val{
          type uint32;
          mandatory true;
          description "operator integer 32 value";
        }
      }
    }
    container operator-option-int16 {
      if-feature operator-op-int16;
      presence "Enable this option";
      description "operator integer 16 option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list int16val{
        key int16val-id;
        description "operator integer 16 info";
        leaf int16val-id {
          type uint8;
          mandatory true;
          description "operator integer 16 id";
        }
        leaf int16val{
          type uint16;
          mandatory true;
          description "operator integer 16 value";
        }
      }
    }
    container operator-option-int8 {
      if-feature operator-op-int8;
      presence "Enable this option";
      description "operator integer 8 option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list int8val{
        key int8val-id;
        description "operator integer 8 info";
        leaf int8val-id {
          type uint8;
          mandatory true;
          description "operator integer 8 id";
```

```
        }
        leaf int8val{
          type uint8;
          mandatory true;
          description "operator integer 8 value";
        }
      }
    }
    container operator-option-uri {
      if-feature operator-op-uri;
      presence "Enable this option";
      description "operator uri option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list uri{
        key uri-id;
        description "operator uri info";
        leaf uri-id {
          type uint8;
          mandatory true;
          description "operator uri id";
        }
        leaf uri{
          type string;
          mandatory true;
          description "operator uri value";
        }
      }
    }
    container operator-option-textstring {
      if-feature operator-op-textstring;
      presence "Enable this option";
      description "operator itext string option";
      reference "RFC7227: Guidelines for Creating New DHCPv6 Options";
      list textstring{
        key textstring-id;
        description "operator text string info";
        leaf textstring-id {
          type uint8;
          mandatory true;
          description "operator text string id";
        }
        leaf textstring{
          type string;
          mandatory true;
          description "operator text string value";
        }
      }
```

```
    }
    container operator-option-var-data {
      if-feature operator-op-var-data;
      presence "Enable this option";
      description "operator variable length data option";
      reference "RFC7227: Guidelines for Creating New DHCPv6 Options";
      list int32val{
        key var-data-id;
        description "operator ivariable length data info";
        leaf var-data-id {
          type uint8;
          mandatory true;
          description "operator variable length id";
        }
        leaf var-data{
          type binary;
          mandatory true;
          description "operator variable length value";
        }
      }
    }
    container operator-option-dns-wire {
      if-feature operator-op-dns-wire;
      presence "Enable this option";
      description "operator dns wire format domain name list option";
      reference "RFC7227: Guidelines for Creating New DHCPv6
        Options";
      list operator-option-dns-wire{
        key operator-option-dns-wire-id;
        description "operator dns wire format info";
        leaf operator-option-dns-wire-id {
          type uint8;
          mandatory true;
          description "operator dns wire format id";
        }
        leaf operator-option-dns-wire{
          type binary;
          mandatory true;
          description "operator dns wire format value";
        }
      }
    }
  }
}

<CODE ENDS>
```

4. Security Considerations (TBD)

TBD

5. IANA Considerations (TBD)

This document registers the following YANG modules in the "YANG Module Names" registry [RFC6020].

```
name:           ietf-dhcpv6
namespace:      urn:ietf:params:xml:ns:yang:ietf-dhcpv6
prefix:         dhcpv6
reference:      TBD
```

6. Acknowledgements

The authors would like to thank Qi Sun, Lishan Li, Sladjana Zoric, Tomek Mrugalski, Marcin Siodelski, Bernie Volz and Bing Liu for their valuable comments and contributions to this work.

7. Contributors

The following individuals contributed to this effort:

Hao Wang
Tsinghua University
Beijing 100084
P.R.China
Phone: +86-10-6278-5822
Email: wangh13@mails.tsinghua.edu.cn

Ted Lemon
Nomium, Inc
950 Charter St.
Redwood City, CA 94043
USA
Email: Ted.Lemon@nomium.com

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <https://www.rfc-editor.org/info/rfc3315>.

[RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, DOI 10.17487/RFC3633, December 2003, <https://www.rfc-editor.org/info/rfc3633>.

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

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <https://www.rfc-editor.org/info/rfc6087>.

[RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <https://www.rfc-editor.org/info/rfc6355>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

8.2. Informative References

[I-D.ietf-netmod-yang-tree-diagrams] Bjorklund, M. and L. Berger, "YANG Tree Diagrams", draft-ietf-netmod-yang-tree-diagrams-04 (work in progress), December 2017.

[RFC3319] Schulzrinne, H. and B. Volz, "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers", RFC 3319, DOI 10.17487/RFC3319, July 2003, <https://www.rfc-editor.org/info/rfc3319>.

[RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <https://www.rfc-editor.org/info/rfc3646>.

[RFC3898] Kalusivalingam, V., "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3898, DOI 10.17487/RFC3898, October 2004, <https://www.rfc-editor.org/info/rfc3898>.

[RFC4075] Kalusivalingam, V., "Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6", RFC 4075, DOI 10.17487/RFC4075, May 2005, <https://www.rfc-editor.org/info/rfc4075>.

[RFC4242] Venaas, S., Chown, T., and B. Volz, "Information Refresh Time Option for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 4242, DOI 10.17487/RFC4242, November 2005, <https://www.rfc-editor.org/info/rfc4242>.

[RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, DOI 10.17487/RFC4704, October 2006, <https://www.rfc-editor.org/info/rfc4704>.

[RFC4833] Lear, E. and P. Eggert, "Timezone Options for DHCP", RFC 4833, DOI 10.17487/RFC4833, April 2007, <https://www.rfc-editor.org/info/rfc4833>.

[RFC5908] Gayraud, R. and B. Lourdelet, "Network Time Protocol (NTP) Server Option for DHCPv6", RFC 5908, DOI 10.17487/RFC5908, June 2010, <https://www.rfc-editor.org/info/rfc5908>.

[RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6 Options for Network Boot", RFC 5970, DOI 10.17487/RFC5970, September 2010, <https://www.rfc-editor.org/info/rfc5970>.

[RFC6334] Hankins, D. and T. Mrugalski, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite", RFC 6334, DOI 10.17487/RFC6334, August 2011, <https://www.rfc-editor.org/info/rfc6334>.

[RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options", RFC 6422, DOI 10.17487/RFC6422, December 2011, <https://www.rfc-editor.org/info/rfc6422>.

[RFC6440] Zorn, G., Wu, Q., and Y. Wang, "The EAP Re-authentication Protocol (ERP) Local Domain Name DHCPv6 Option", RFC 6440, DOI 10.17487/RFC6440, December 2011, <https://www.rfc-editor.org/info/rfc6440>.

[RFC6784] Sakane, S. and M. Ishiyama, "Kerberos Options for DHCPv6", RFC 6784, DOI 10.17487/RFC6784, November 2012, <https://www.rfc-editor.org/info/rfc6784>.

[RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer Address Option in DHCPv6", RFC 6939, DOI 10.17487/RFC6939, May 2013, <https://www.rfc-editor.org/info/rfc6939>.

[RFC7078] Matsumoto, A., Fujisaki, T., and T. Chown, "Distributing Address Selection Policy Using DHCPv6", RFC 7078, DOI 10.17487/RFC7078, January 2014, <https://www.rfc-editor.org/info/rfc7078>.

[RFC7083] Droms, R., "Modification to Default Values of SOL_MAX_RT and INF_MAX_RT", RFC 7083, DOI 10.17487/RFC7083, November 2013, <https://www.rfc-editor.org/info/rfc7083>.

[RFC7227] Hankins, D., Mrugalski, T., Siodelski, M., Jiang, S., and S. Krishnan, "Guidelines for Creating New DHCPv6 Options", BCP 187, RFC 7227, DOI 10.17487/RFC7227, May 2014, <https://www.rfc-editor.org/info/rfc7227>.

[RFC7291] Boucadair, M., Penno, R., and D. Wing, "DHCP Options for the Port Control Protocol (PCP)", RFC 7291, DOI 10.17487/RFC7291, July 2014, <https://www.rfc-editor.org/info/rfc7291>.

[RFC7598] Mrugalski, T., Troan, O., Farrer, I., Perreault, S., Dec, W., Bao, C., Yeh, L., and X. Deng, "DHCPv6 Options for Configuration of Softwire Address and Port-Mapped Clients", RFC 7598, DOI 10.17487/RFC7598, July 2015, <https://www.rfc-editor.org/info/rfc7598>.

Authors' Addresses

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

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

Linhui Sun
Tsinghua University
Beijing 100084
P.R.China

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

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

Email: ian.farrer@telekom.de

Sladjana Zechlin
Deutsche Telekom AG
CTO-IPT, Landgrabenweg 151
Bonn, NRW 53227
Germany

Email: sladjana.zechlin@telekom.de

Zihao He
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-10-6278-5822
Email: hezihao9512@gmail.com