DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 December 2021

I. Farrer, Ed.
Deutsche Telekom AG

1 June 2021

YANG Data Model for DHCPv6 Configuration
draft-ietf-dhc-dhcpv6-yang-20

Abstract

This document describes YANG data modules for the configuration and management of DHCPv6 (Dynamic Host Configuration Protocol for IPv6) servers, relays, and clients.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

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 3 December 2021.

Table of Contents

1. Introduction

DHCPv6 [RFC8415] is widely used for supplying configuration and other relevant parameters to clients in IPv6 networks. This document defines YANG [RFC7950] modules for the configuration and management of DHCPv6 'element' (servers, relays, and clients) using the Network Configuration Protocol (NETCONF [RFC6241]) or RESTCONF [RFC8040] protocols.

Separate modules are defined for each element. Additionally, a 'common' module contains typedefs and groupings used by all of the element modules. Appendix A provides XML examples for each of the element modules and shows their interaction.

The relay and client modules provide configuration which is applicable device's interfaces. This is done by importing the ietf-interfaces module [RFC8343] and using interface-refs to the relevant interface(s).

It is worth noting that as DHCPv6 is itself a client configuration protocol, it is not the intention of this document to provide a replacement for the allocation of DHCPv6 assigned addressing and parameters by using NETCONF/YANG. The DHCPv6 client module is intended for the configuration and monitoring of the DHCPv6 client function and does not replace DHCPv6 address and parameter configuration.

The YANG modules in this document adopt the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Scope

[RFC8415] describes the current version of the DHCPv6 base protocol specification. A large number of additional specifications have also been published, extending DHCPv6 element functionality and adding new options. The YANG modules contained in this document do not attempt to capture all of these extensions and additions, rather to model the DHCPv6 functions and options covered in [RFC8415]. A focus has also been given on the extensibility of the modules so that they are easy to augment to add additional functionality as required by a particular implementation or deployment scenario.

1.2. Extensibility of the DHCPv6 Server YANG Module

The modules in this document only attempt to model DHCPv6-specific behavior and do not cover the configuration and management of functionality relevant for specific server implementations. The level of variance between implementations is too great to attempt to standardize them in a way that is useful without being restrictive.

However, it is recognized that implementation-specific configuration and management is also an essential part of DHCP deployment and operations. To resolve this, Appendix C contains an example YANG module for the configuration of implementation-specific functions, illustrating how this functionality can be augmented into the main 'ietf-dhcpv6-server.yang' module.

In DHCPv6, the concept of 'class selection' for messages received by the server is common. This is the identification and classification of messages based on a number of parameters so that the correct provisioning information can be supplied. For example, allocating a prefix from the correct pool, or supplying a set of options relevant for a specific vendor's client implementation. During the development of this document, implementations were researched and the findings were that while this function is common to all, the method for configuring and implementing this function differs greatly. Therefore, configuration of the class selection function has been omitted from the DHCPv6 server module to allow implementors to define their own suitable YANG modules. Appendix D provides an example of this, to demonstrate how this is can be integrated with the main 'ietf-dhcpv6-server.yang' module.

1.2.1. DHCPv6 Option Definitions

A large number of DHCPv6 options have been created in addition to those defined in [RFC8415]. As implementations differ widely as to which DHCPv6 options they support, the following approach has been taken to defining options: Only the DHCPv6 options defined in [RFC8415] are included in this document.

Of these, only the options that require operator configuration are modelled. For example, OPTION_IA_NA (3) is created by the DHCP server when requested by the client. The contents of the fields in the option are based on a number of input configuration parameters which the server will apply when it receives the request (e.g., the T1/T2 timers that are relevant for the pool of addresses). As a result, there are no fields that are directly configurable for the option, so it is not modelled.

The following table shows the DHCPv6 options that are modeled, the element(s) they are sent by, and the relevant YANG module name:

Name	Server	Relay	Client	Module Name
OPTION_ORO (6) Option Request Option			X	ietf-dhcpv6-client.yang
OPTION_PREFERENCE (7) Preference Option	X			ietf-dhcpv6-server.yang
OPTION_AUTH (11) Authentication Option	X	X		ietf-dhcpv6-common.yang
OPTION_UNICAST (12) Server Unicast Option	X			ietf-dhcpv6-server.yang
OPTION_STATUS_CODE (13) Status Code Option	X	X	X	ietf-dhcpv6-common.yang
OPTION_RAPID_COMMIT (14) Rapid Commit Option	X		X	ietf-dhcpv6-common.yang
OPTION_USER_CLASS (15) User Class Option			X	ietf-dhcpv6-client.yang
OPTION_VENDOR_CLASS (16) Vendor Class Option			X	ietf-dhcpv6-client.yang
OPTION_VENDOR_OPTS (17) Vendor-specific Information Option	X		X	ietf-dhcpv6-common.yang
OPTION_INTERFACE_ID (18) Interface-Id Option		X		ietf-dhcpv6-relay.yang
OPTION_RECONF_MSG (19) Reconfigure Message Option	X			ietf-dhcpv6-server.yang
OPTION_RECONF_ACCEPT (20) Reconfigure	X		X	ietf-dhcpv6-client.yang

Accept Option				
OPTION_INFORMATION_REFRESH_TIME (32) Information Refresh Time Option	X			ietf-dhcpv6-server.yang
OPTION_SOL_MAX_RT (82) sol max rt Option	X			ietf-dhcpv6-server.yang
OPTION_INF_MAX_RT (83) inf max rt Option	X			ietf-dhcpv6-server.yang

Table 1: Modeled DHCPv6 Options

Further options definitions can be added using additional YANG modules via augmentation of the relevant element modules from this document. Appendix B contains an example module showing how the DHCPv6 option definitions can be extended in this manner. Some guidance on how to write YANG modules for additional DHCPv6 options is also provided.

1.3. Terminology

The reader should be familiar with the YANG data modeling language defined in [RFC7950].

The YANG modules in this document adopt the Network Management Datastore Architecture (NMDA) [RFC8342]. The meanings of the symbols used in tree diagrams are defined in [RFC8340].

The reader should be familiar with DHCPv6 relevant terminology as defined in [RFC8415] and other relevant documents.

2. DHCPv6 Tree Diagrams

2.1. DHCPv6 Server Tree Diagram

The tree diagram in Figure 1 provides an overview of the DHCPv6 server module. The tree also includes the common functions module Section 3.4.

```
module: ietf-dhcpv6-server
  +--rw dhcpv6-server
     +--rw enabled?          boolean
     +--rw server-duid?      dhc6:duid
     +--rw vendor-config
     +--rw option-sets
     |  +--rw option-set* [option-set-id]
     |     +--rw option-set-id                          uint32
     |     +--rw description?                           string
     |     +--rw preference-option
     |     |  +--rw pref-value?   uint8
     |     +--rw auth-option
     |     |  +--rw protocol?           uint8
     |     |  +--rw algorithm?          uint8
     |     |  +--rw rdm?                uint8
     |     |  +--rw replay-detection?   uint64
     |     |  +--rw auth-information?   string
     |     +--rw server-unicast-option
     |     |  +--rw server-address?   inet:ipv6-address
     |     +--rw status-code-option
     |     |  +--rw status-code?      uint16
     |     |  +--rw status-message?   string
     |     +--rw rapid-commit-option!
     |     +--rw vendor-specific-information-options
     |     |  +--rw vendor-specific-information-option*
     |     |           [enterprise-number]
     |     |     +--rw enterprise-number     uint32
     |     |     +--rw vendor-option-data* [sub-option-code]
     |     |        +--rw sub-option-code    uint16
     |     |        +--rw sub-option-data?   string
     |     +--rw reconfigure-message-option
     |     |  +--rw msg-type?   uint8
     |     +--rw reconfigure-accept-option!
     |     +--rw info-refresh-time-option
     |     |  +--rw info-refresh-time?   dhc6:timer-seconds32
     |     +--rw sol-max-rt-option
     |     |  +--rw sol-max-rt-value?   dhc6:timer-seconds32
     |     +--rw inf-max-rt-option
     |        +--rw inf-max-rt-value?   dhc6:timer-seconds32
     +--rw class-selector
     +--rw network-ranges
        +--rw option-set-id*              leafref
        +--rw valid-lifetime?             dhc6:timer-seconds32
        +--rw renew-time?                 dhc6:timer-seconds32
        +--rw rebind-time?                dhc6:timer-seconds32
        +--rw preferred-lifetime?         dhc6:timer-seconds32
        +--rw rapid-commit?               boolean
        +--rw network-range* [id]
```

```
|  +--rw id                    uint32
|  +--rw description?          string
|  +--rw network-prefix        inet:ipv6-prefix
|  +--rw option-set-id*        leafref
|  +--rw valid-lifetime?       dhc6:timer-seconds32
|  +--rw renew-time?           dhc6:timer-seconds32
|  +--rw rebind-time?          dhc6:timer-seconds32
|  +--rw preferred-lifetime?   dhc6:timer-seconds32
|  +--rw rapid-commit?         boolean
|  +--rw address-pools
|  |  +--rw address-pool* [pool-id]
|  |     +--rw pool-id                    uint32
|  |     +--rw pool-prefix
|  |     |       inet:ipv6-prefix
|  |     +--rw start-address
|  |     |       inet:ipv6-address-no-zone
|  |     +--rw end-address
|  |     |       inet:ipv6-address-no-zone
|  |     +--rw max-address-utilization?   dhc6:threshold
|  |     +--rw option-set-id*             leafref
|  |     +--rw valid-lifetime?
|  |     |       dhc6:timer-seconds32
|  |     +--rw renew-time?
|  |     |       dhc6:timer-seconds32
|  |     +--rw rebind-time?
|  |     |       dhc6:timer-seconds32
|  |     +--rw preferred-lifetime?
|  |     |       dhc6:timer-seconds32
|  |     +--rw rapid-commit?              boolean
|  |     +--rw host-reservations
|  |     |  +--rw host-reservation* [reserved-addr]
|  |     |     +--rw client-duid?          dhc6:duid
|  |     |     +--rw reserved-addr
|  |     |     |       inet:ipv6-address
|  |     |     +--rw option-set-id*        leafref
|  |     |     +--rw valid-lifetime?
|  |     |     |       dhc6:timer-seconds32
|  |     |     +--rw renew-time?
|  |     |     |       dhc6:timer-seconds32
|  |     |     +--rw rebind-time?
|  |     |     |       dhc6:timer-seconds32
|  |     |     +--rw preferred-lifetime?
|  |     |     |       dhc6:timer-seconds32
|  |     |     +--rw rapid-commit?         boolean
|  |     +--ro active-leases
|  |        +--ro total-count        uint64
|  |        +--ro allocated-count    uint64
|  |        +--ro active-lease* [leased-address]
```

```
|  |          +--ro leased-address
|  |          |       inet:ipv6-address
|  |          +--ro client-duid?          dhc6:duid
|  |          +--ro ia-id                 uint32
|  |          +--ro allocation-time?
|  |          |       yang:date-and-time
|  |          +--ro last-renew-rebind?
|  |          |       yang:date-and-time
|  |          +--ro preferred-lifetime?
|  |          |       dhc6:timer-seconds32
|  |          +--ro valid-lifetime?
|  |          |       dhc6:timer-seconds32
|  |          +--ro lease-t1?
|  |          |       dhc6:timer-seconds32
|  |          +--ro lease-t2?
|  |                  dhc6:timer-seconds32
|  +--rw prefix-pools {prefix-delegation}?
|     +--rw prefix-pool* [pool-id]
|        +--rw pool-id                     uint32
|        +--rw pool-prefix
|        |       inet:ipv6-prefix
|        +--rw client-prefix-length        uint8
|        +--rw max-pd-space-utilization?   dhc6:threshold
|        +--rw option-set-id*              leafref
|        +--rw valid-lifetime?
|        |       dhc6:timer-seconds32
|        +--rw renew-time?
|        |       dhc6:timer-seconds32
|        +--rw rebind-time?
|        |       dhc6:timer-seconds32
|        +--rw preferred-lifetime?
|        |       dhc6:timer-seconds32
|        +--rw rapid-commit?               boolean
|        +--rw host-reservations
|        |  +--rw prefix-reservation* [reserved-prefix]
|        |  |  +--rw client-duid?           dhc6:duid
|        |  |  +--rw reserved-prefix
|        |  |  |       inet:ipv6-prefix
|        |  |  +--rw reserved-prefix-len?   uint8
|        |  +--rw option-set-id*        leafref
|        |  +--rw valid-lifetime?
|        |  |       dhc6:timer-seconds32
|        |  +--rw renew-time?
|        |  |       dhc6:timer-seconds32
|        |  +--rw rebind-time?
|        |  |       dhc6:timer-seconds32
|        |  +--rw preferred-lifetime?
|        |  |       dhc6:timer-seconds32
```

```
          |        |  +--rw rapid-commit?         boolean
          |        +--ro active-leases
          |           +--ro total-count        uint64
          |           +--ro allocated-count    uint64
          |           +--ro active-lease* [leased-prefix]
          |              +--ro leased-prefix
          |              |       inet:ipv6-prefix
          |              +--ro client-duid?          dhc6:duid
          |              +--ro ia-id                 uint32
          |              +--ro allocation-time?
          |              |       yang:date-and-time
          |              +--ro last-renew-rebind?
          |              |       yang:date-and-time
          |              +--ro preferred-lifetime?
          |              |       dhc6:timer-seconds32
          |              +--ro valid-lifetime?
          |              |       dhc6:timer-seconds32
          |              +--ro lease-t1?
          |              |       dhc6:timer-seconds32
          |              +--ro lease-t2?
          |                      dhc6:timer-seconds32
          +--ro solicit-count?               uint32
          +--ro advertise-count?             uint32
          +--ro request-count?               uint32
          +--ro confirm-count?               uint32
          +--ro renew-count?                 uint32
          +--ro rebind-count?                uint32
          +--ro reply-count?                 uint32
          +--ro release-count?               uint32
          +--ro decline-count?               uint32
          +--ro reconfigure-count?           uint32
          +--ro information-request-count?   uint32

      rpcs:
        +---x delete-address-lease
        |  +---w input
        |  |  +---w lease-address-to-delete    leafref
        |  +--ro output
        |     +--ro return-message?   string
        +---x delete-prefix-lease {prefix-delegation}?
           +---w input
           |  +---w lease-prefix-to-delete    leafref
           +--ro output
              +--ro return-message?   string

      notifications:
        +---n address-pool-utilization-threshold-exceeded
        |  +--ro pool-id                   leafref
```

```
        |  +--ro total-pool-addresses       uint64
        |  +--ro max-allocated-addresses    uint64
        |  +--ro allocated-address-count    uint64
        +---n prefix-pool-utilization-threshold-exceeded
        |       {prefix-delegation}?
        |  +--ro pool-id                     leafref
        |  +--ro total-pool-prefixes         uint64
        |  +--ro max-allocated-prefixes      uint64
        |  +--ro allocated-prefixes-count    uint64
        +---n invalid-client-detected
        |  +--ro message-type?   enumeration
        |  +--ro duid?           dhc6:duid
        |  +--ro description?    string
        +---n decline-received
        |  +--ro duid?                 dhc6:duid
        |  +--ro declined-resources* []
        |     +--ro (resource-type)?
        |        +--:(declined-address)
        |        |  +--ro address?   inet:ipv6-address
        |        +--:(declined-prefix)
        |           +--ro prefix?    inet:ipv6-prefix
        +---n non-success-code-sent
           +--ro status-code    uint16
           +--ro duid?          dhc6:duid
```

Figure 1: DHCPv6 Server Data Module Structure

Descriptions of important nodes:

* enabled: Enables/disables the function of the DHCPv6 server.

* dhcpv6-server: This container holds the server's DHCPv6 specific configuration.

* server-duid: Each server must have a DUID (DHCP Unique Identifier) to identify itself to clients. A DUID consists of a two-octet type field and an arbitrary length (of no more than 128-octets) content field. Currently there are four defined types of DUIDs in [RFC8415] and [RFC6355]. The DUID may be configured using the format for one of these types, or using the 'unstructured' format. The DUID type definitions are imported from the 'ietf-dhcpv6-common.yang' module. [IANA-HARDWARE-TYPES] and [IANA-PEN] are referenced for the relevant DUID types.

* vendor-config: This container is provided as a location for additional implementation-specific YANG nodes for the configuration of the device to be augmented. See Appendix C for an example of such a module.

* option-sets: The server can be configured with multiple option-sets. These are groups of DHCPv6 options with common parameters which will be supplied to clients on request. The 'option-set-id' field is used to reference an option-set elsewhere in the server's configuration.

* option-set: Holds configuration parameters for DHCPv6 options. The initial set of applicable option definitions are defined here and additional options that are also relevant to the relay and/or client are imported from the 'ietf-dhcpv6-common' module. Where needed, other DHCPv6 option modules can be augmented as they are defined.

* class-selector: This is provided as a location for additional implementation specific YANG nodes for vendor specific class selector nodes to be augmented. See Appendix D for an example of this.

* network-ranges: A hierarchical model is used for the allocation of addresses and prefixes. At the top level, 'network-ranges' holds global configuration parameters. Under this, a list of 'network-ranges' can be defined. Inside 'network-rages', 'address-pools' (for IA_NA and IA_TA allocations), and 'prefix-pools' (for IA_PD allocation) are defined. Finally within the pools, specific host-reservations are held.

* prefix-pools: Defines pools to be used for prefix delegation to clients. As prefix delegation is not supported by all DHCPv6 server implementations, it is enabled by a feature statement.

Information about notifications:

* address/prefix-pool-utilization-threshold-exceeded: Raised when the number of leased addresses or prefixes exceeds the configured usage threshold.

* invalid-client-detected: Raised when the server detects an invalid client. A description of the error and message type that has generated the notification can be included.

* decline-received: Raised when a DHCPv6 Decline message is received from a client.

* non-success-code-sent: Raised when a status message is raised for an error.

Information about RPCs

* delete-address-lease: Allows the deletion of a lease for an individual IPv6 address from the server's lease database.

* delete-prefix-lease: Allows the deletion of a lease for an individual IPv6 prefix from the server's lease database.

2.2. DHCPv6 Relay Tree Diagram

The tree diagram in Figure 2 provides an overview of the DHCPv6 relay module. The tree also includes the common functions module Section 3.4.

```
    module: ietf-dhcpv6-relay
      +--rw dhcpv6-relay
         +--rw enabled?                                  boolean
         +--rw relay-if* [if-name]
         |  +--rw if-name
         |  |       if:interface-ref
         |  +--rw enabled?                                boolean
         |  +--rw destination-address*
         |  |       inet:ipv6-address
         |  +--rw link-address?
         |  |       inet:ipv6-address
         |  +--rw relay-options
         |  |  +--rw auth-option
         |  |  |  +--rw protocol?           uint8
         |  |  |  +--rw algorithm?          uint8
         |  |  |  +--rw rdm?                uint8
         |  |  |  +--rw replay-detection?   uint64
         |  |  |  +--rw auth-information?   string
         |  |  +--rw status-code-option
         |  |  |  +--rw status-code?      uint16
         |  |  |  +--rw status-message?   string
         |  |  +--rw interface-id-option
         |  |     +--rw interface-id?   string
         |  +--ro solicit-received-count?               uint32
         |  +--ro advertise-sent-count?                 uint32
         |  +--ro request-received-count?               uint32
         |  +--ro confirm-received-count?               uint32
         |  +--ro renew-received-count?                 uint32
         |  +--ro rebind-received-count?                uint32
         |  +--ro reply-sent-count?                     uint32
         |  +--ro release-received-count?               uint32
         |  +--ro decline-received-count?               uint32
         |  +--ro reconfigure-sent-count?               uint32
         |  +--ro information-request-received-count?   uint32
         |  +--ro unknown-message-received-count?       uint32
         |  +--ro unknown-message-sent-count?           uint32
```

```
          |  +--ro discarded-message-count?             uint32
          |  +--rw prefix-delegation! {prefix-delegation}?
          |     +--ro pd-leases* [ia-pd-prefix]
          |        +--ro ia-pd-prefix          inet:ipv6-prefix
          |        +--ro last-renew?           yang:date-and-time
          |        +--ro client-peer-address?  inet:ipv6-address
          |        +--ro client-duid?          dhc6:duid
          |        +--ro server-duid?          dhc6:duid
          +--ro relay-forward-sent-count?              uint32
          +--ro relay-forward-received-count?          uint32
          +--ro relay-reply-received-count?            uint32
          +--ro relay-forward-unknown-sent-count?      uint32
          +--ro relay-forward-unknown-received-count?  uint32
          +--ro discarded-message-count?               uint32

       rpcs:
         +---x clear-prefix-entry {prefix-delegation}?
         |  +---w input
         |  |  +---w lease-prefix    leafref
         |  +--ro output
         |     +--ro return-message?   string
         +---x clear-client-prefixes {prefix-delegation}?
         |  +---w input
         |  |  +---w client-duid    dhc6:duid
         |  +--ro output
         |     +--ro return-message?   string
         +---x clear-interface-prefixes {prefix-delegation}?
            +---w input
            |  +---w interface
            |          -> ../../dhcpv6-relay/relay-if/if-name
            +--ro output
               +--ro return-message?   string

       notifications:
         +---n relay-event
            +--ro topology-change
               +--ro relay-if-name?
               |       -> /dhcpv6-relay/relay-if/if-name
               +--ro last-ipv6-addr?   inet:ipv6-address
```

Figure 2: DHCPv6 Relay Data Module Structure

Descriptions of important nodes:

* enabled: Globally enables/disables all DHCPv6 relay functions.

* dhcpv6-relay: This container holds the relay's DHCPv6-specific configuration.

* relay-if: As a relay may have multiple client-facing interfaces, they are configured in a list. The if-name leaf is the key and is an interface-ref to the applicable interface defined by the 'ietf-interfaces' YANG module.

* enabled: Enables/disables all DHCPv6 relay function for the specific interface.

* destination-addresses: Defines a list of IPv6 addresses that client messages will be relayed to. May include unicast or multicast addresses.

* link-address: Configures the value that the relay will put into the link-address field of Relay-Forward messages.

* prefix-delegation: As prefix delegation is not supported by all DHCPv6 relay implementations, it is enabled by this feature statement where required.

* pd-leases: Contains read-only nodes for holding information about active delegated prefix leases.

* relay-options: Holds configuration parameters for DHCPv6 options which can be sent by the relay. The initial set of applicable option definitions are defined here and additional options that are also relevant to the server and/or client are imported from the 'ietf-dhcpv6-common' module. Where needed, other DHCPv6 option modules can be augmented as they are defined.

Information about notifications:

* topology-changed: Raised when the topology of the relay agent is changed, e.g., a client facing interface is reconfigured.

Information about RPCs

* clear-prefix-lease: Allows the removal of a delegated lease entry from the relay.

* clear-client-prefixes: Allows the removal of all of the delegated lease entries for a single client (referenced by client DUID) from the relay.

* clear-interface-prefixes: Allows the removal of all of the delegated lease entries from an interface on the relay.

2.3. DHCPv6 Client Tree Diagram

The tree diagram in Figure 3 provides an overview of the DHCPv6 client module. The tree also includes the common functions module defined in Section 3.4.

```
module: ietf-dhcpv6-client
  +--rw dhcpv6-client
     +--rw enabled?     boolean
     +--rw client-if* [if-name]
        +--rw if-name                       if:interface-ref
        +--rw enabled?                      boolean
        +--rw duid?                         dhc6:duid
        +--rw client-configured-options
        |  +--rw option-request-option
        |  |  +--rw oro-option*   uint16
        |  +--rw status-code-option
        |  |  +--rw status-code?      uint16
        |  |  +--rw status-message?   string
        |  +--rw rapid-commit-option!
        |  +--rw user-class-option!
        |  |  +--rw user-class-data-instance*
        |  |          [user-class-data-id]
        |  |     +--rw user-class-data-id    uint8
        |  |     +--rw user-class-data?      string
        |  +--rw vendor-class-option
        |  |  +--rw vendor-class-option-instances*
        |  |          [enterprise-number]
        |  |     +--rw enterprise-number            uint32
        |  |     +--rw vendor-class-data-element*
        |  |             [vendor-class-data-id]
        |  |        +--rw vendor-class-data-id    uint8
        |  |        +--rw vendor-class-data?      string
        |  +--rw vendor-specific-information-options
        |  |  +--rw vendor-specific-information-option*
        |  |          [enterprise-number]
        |  |     +--rw enterprise-number     uint32
        |  |     +--rw vendor-option-data* [sub-option-code]
        |  |        +--rw sub-option-code    uint16
        |  |        +--rw sub-option-data?   string
        |  +--rw reconfigure-accept-option!
        +--rw ia-na* [ia-id]
        |  +--rw ia-id            uint32
        |  +--rw ia-na-options
        |  +--ro lease-state
        |     +--ro ia-na-address?        inet:ipv6-address
        |     +--ro preferred-lifetime?   dhc6:timer-seconds32
        |     +--ro valid-lifetime?       dhc6:timer-seconds32
```

```
          |     +--ro lease-t1?             dhc6:timer-seconds32
          |     +--ro lease-t2?             dhc6:timer-seconds32
          |     +--ro allocation-time?      yang:date-and-time
          |     +--ro last-renew-rebind?    yang:date-and-time
          |     +--ro server-duid?          dhc6:duid
          +--rw ia-ta* [ia-id]
          |  +--rw ia-id            uint32
          |  +--rw ia-ta-options
          |  +--ro lease-state
          |     +--ro ia-ta-address?        inet:ipv6-address
          |     +--ro preferred-lifetime?   dhc6:timer-seconds32
          |     +--ro valid-lifetime?       dhc6:timer-seconds32
          |     +--ro allocation-time?      yang:date-and-time
          |     +--ro last-renew-rebind?    yang:date-and-time
          |     +--ro server-duid?          dhc6:duid
          +--rw ia-pd* [ia-id]
          |  +--rw ia-id            uint32
          |  +--rw ia-pd-options
          |  +--ro lease-state
          |     +--ro ia-pd-prefix?         inet:ipv6-prefix
          |     +--ro preferred-lifetime?   dhc6:timer-seconds32
          |     +--ro valid-lifetime?       dhc6:timer-seconds32
          |     +--ro lease-t1?             dhc6:timer-seconds32
          |     +--ro lease-t2?             dhc6:timer-seconds32
          |     +--ro allocation-time?      yang:date-and-time
          |     +--ro last-renew-rebind?    yang:date-and-time
          |     +--ro server-duid?          dhc6:duid
          +--ro solicit-count?               uint32
          +--ro advertise-count?             uint32
          +--ro request-count?               uint32
          +--ro confirm-count?               uint32
          +--ro renew-count?                 uint32
          +--ro rebind-count?                uint32
          +--ro reply-count?                 uint32
          +--ro release-count?               uint32
          +--ro decline-count?               uint32
          +--ro reconfigure-count?           uint32
          +--ro information-request-count?   uint32

      notifications:
        +---n invalid-ia-address-detected
        |  +--ro ia-id                 uint32
        |  +--ro ia-na-t1-timer?       uint32
        |  +--ro ia-na-t2-timer?       uint32
        |  +--ro invalid-address?      inet:ipv6-address
        |  +--ro preferred-lifetime?   uint32
        |  +--ro valid-lifetime?       uint32
        |  +--ro ia-options?           binary
```

```
         |  +--ro description?          string
         +---n transmission-failed
         |  +--ro failure-type    enumeration
         |  +--ro description?    string
         +---n unsuccessful-status-code
         |  +--ro status-code    uint16
         |  +--ro server-duid    dhc6:duid
         +---n server-duid-changed
            +--ro new-server-duid         dhc6:duid
            +--ro previous-server-duid    dhc6:duid
            +--ro lease-ia-na?
            |       -> /dhcpv6-client/client-if/ia-na/ia-id
            +--ro lease-ia-ta?
            |       -> /dhcpv6-client/client-if/ia-ta/ia-id
            +--ro lease-ia-pd?
                    -> /dhcpv6-client/client-if/ia-pd/ia-id
```

Figure 3: DHCPv6 Client Data Module Structure

Descriptions of important nodes:

* enabled: Globally enables/disables all DHCPv6 client functions.

* dhcpv6-client: This container holds the client's DHCPv6 specific configuration.

* client-if: As a client may have multiple interfaces requesting configuration over DHCP, they are configured in a list. The if-name leaf is the key and is an interface-ref to the applicable interface defined by the 'ietf-interfaces' YANG module.

* enabled: Enables/disables all DHCPv6 client function for the specific interface.

* client-duid: Each client must have a DUID (DHCP Unique Identifier) to identify itself to servers and relays. A DUID consists of a two-octet type field and an arbitrary length (1-128 octets) content field. Currently there are four defined types of DUIDs in [RFC8415] and [RFC6355]. The DUID may be configured using the format for one of these types, or using the 'unstructured' format. The DUID type definitions are imported from the 'ietf-dhcpv6-common.yang' module. [IANA-HARDWARE-TYPES] and [IANA-PEN] are referenced for the relevant DUID types.

* client-configured-options: Holds configuration parameters for DHCPv6 options which can be sent by the client. The initial set of applicable option definitions are defined here and additional options that are also relevant to the relay and/or server are imported from the 'ietf-dhcpv6-common' module. Where needed, other DHCPv6 option modules can be augmented as they are defined.

* ia-na, ia-ta, ia-pd: Contains configuration nodes relevant for requesting one or more of each of the lease types. Read-only nodes related to the active leases for each type are also located here.

Information about notifications:

* invalid-ia-detected: Raised when the identity association of the client can be proved to be invalid. Possible conditions include: duplicated address, illegal address, etc.

* retransmission-failed: Raised when the retransmission mechanism defined in [RFC8415] has failed.

3. DHCPv6 YANG Modules

3.1. DHCPv6 Server YANG Module

This module imports typedefs from [RFC6991], [RFC8343].

<CODE BEGINS> file "ietf-dhcpv6-server@2021-06-01.yang"

```
module ietf-dhcpv6-server {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server";
  prefix "dhc6-srv";

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix yang;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-dhcpv6-common {
    prefix dhc6;
```

```
    reference
      "RFC XXXX: To be updated on publication";
  }

  import ietf-netconf-acm {
    prefix nacm;
    reference
      "RFC 8341: Network Configuration Access Control Model";
  }

  organization
    "IETF DHC (Dynamic Host Configuration) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
     WG List:  <mailto:dhcwg@ietf.org>
     Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
     Author:   Linhui Sun <lh.sunlinh@gmail.com>
     Editor:   Ian Farrer <ian.farrer@telekom.de>
     Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
     Author:   Zihao He <hezihao9512@gmail.com>
     Author:   Michal Nowikowski <godfryd@isc.org>";

  description
    "This YANG module defines components for the configuration
    and management of DHCPv6 servers.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";

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

  /*
   * Features
```

```
     */

    feature prefix-delegation {
      description
        "Denotes that the server implements DHCPv6 prefix
        delegation.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 6.3";
    }

    /*
     * Groupings
     */

    grouping resource-config {
      description
      "Nodes that are reused at multiple levels in the DHCPv6
      server's addressing hierarchy.";
      leaf-list option-set-id {
        type leafref {
          path "/dhcpv6-server/option-sets/option-set/option-set-id";
        }
        description
          "The ID field of relevant set of DHCPv6 options (option-set)
          to be provisioned to clients using the network-range.";
      }
      leaf valid-lifetime {
        type dhc6:timer-seconds32;
        description
          "Valid lifetime for the Identity Association (IA).";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 6";
      }
      leaf renew-time {
        type dhc6:timer-seconds32;
        description
          "Renew (T1) time.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 4.2";
      }
      leaf rebind-time {
        type dhc6:timer-seconds32;
        description
          "Rebind (T2) time.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 4.2";
      }
      leaf preferred-lifetime {
```

```
        type dhc6:timer-seconds32;
        description
          "Preferred lifetime for the Identity Association (IA).";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 6";
      }
      leaf rapid-commit {
        type boolean;
        description
          "When set to 'true', Specifies that the pool supports
          client-server exchanges involving two messages.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 5.1";
      }
    }

    grouping lease-information {
      description
        "Binding information for each client that has been allocated
        an IPv6 address or prefix.";
      leaf client-duid {
        type dhc6:duid;
        description
          "Client DUID.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 11";
      }
      leaf ia-id {
        type uint32;
        mandatory true;
        description
          "Client's IAID";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 12";
      }
      leaf allocation-time {
        type yang:date-and-time;
        description
          "Time and date that the lease was made.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 18";
      }
      leaf last-renew-rebind {
        type yang:date-and-time;
        description
          "Time of the last successful renew or rebind.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 18";
```

```
      }
      leaf preferred-lifetime {
        type dhc6:timer-seconds32;
        description
          "The preferred lifetime expressed in seconds.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 6";
      }
      leaf valid-lifetime {
        type dhc6:timer-seconds32;
        description
          "The valid lifetime for the lease expressed in seconds.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 6";
      }
      leaf lease-t1 {
        type dhc6:timer-seconds32;
        description
          "The time interval after which the client should contact
          the server from which the addresses in the IA_NA were
          obtained to extend the lifetimes of the addresses assigned
          to the IA_PD.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 4.2";
      }
      leaf lease-t2 {
        type dhc6:timer-seconds32;
        description
          "The time interval after which the client should contact
          any available server to extend the lifetimes of the
          addresses assigned to the IA_PD.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 4.2";
      }
    }

    grouping message-stats {
      description
        "Counters for DHCPv6 messages.";
      leaf solicit-count {
        type uint32;
        config "false";
        description
          "Number of Solicit (1) messages received.";
      }
      leaf advertise-count {
        type uint32;
        config "false";
```

```
        description
          "Number of Advertise (2) messages sent.";
      }
      leaf request-count {
        type uint32;
        config "false";
        description
          "Number of Request (3) messages received.";
      }
      leaf confirm-count {
        type uint32;
        config "false";
        description
          "Number of Confirm (4) messages received.";
      }
      leaf renew-count {
        type uint32;
        config "false";
        description
          "Number of Renew (5) messages received.";
      }
      leaf rebind-count {
        type uint32;
        config "false";
        description
          "Number of Rebind (6) messages received.";
      }
      leaf reply-count {
        type uint32;
        config "false";
        description
          "Number of Reply (7) messages sent.";
      }
      leaf release-count {
        type uint32;
        config "false";
        description
          "Number of Release (8) messages received.";
      }
      leaf decline-count {
        type uint32;
        config "false";
        description
          "Number of Decline (9) messages received.";
      }
      leaf reconfigure-count {
        type uint32;
        config "false";
```

```
        description
          "Number of Reconfigure (10) messages sent.";
      }
      leaf information-request-count {
        type uint32;
        config "false";
        description
          "Number of Information-request (11) messages
          received.";
      }
    }

    grouping preference-option-group {
      description
        "OPTION_PREFERENCE (7) Preference Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.8";
     container preference-option {
       description
         "OPTION_PREFERENCE (7) Preference Option
         container.";
       leaf pref-value {
          type uint8;
          description
            "The preference value for the server in this
            message. A 1-octet unsigned integer.";
        }
      }
    }

    grouping server-unicast-option-group {
      description
        "OPTION_UNICAST (12) Server Unicast Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.12";
      container server-unicast-option {
        description
          "OPTION_UNICAST (12) Server Unicast Option container.";
        leaf server-address {
          type inet:ipv6-address;
          description
            "The 128-bit address to which the client should send
            messages delivered using unicast.";
        }
      }
    }

    grouping reconfigure-message-option-group {
```

```
      description
        "OPTION_RECONF_MSG (19) Reconfigure Message Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.19";
      container reconfigure-message-option {
        description
          "OPTION_RECONF_MSG (19) Reconfigure Message Option.";
        leaf msg-type {
          type uint8;
          description
            "5 for Renew message, 6 for Rebind message, 11 for
            Information-request message.";
        }
      }
    }

    grouping info-refresh-time-option-group {
      description
        "OPTION_INFORMATION_REFRESH_TIME (32) Information Refresh
        Time Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.23";
      container info-refresh-time-option {
        description
          "OPTION_INFORMATION_REFRESH_TIME (32)
          Information Refresh Time option container.";
        leaf info-refresh-time {
          type dhc6:timer-seconds32;
          description
            "Time duration relative to the current time, expressed
            in units of seconds.";
        }
      }
    }

    grouping sol-max-rt-option-group {
      description
        "OPTION_SOL_MAX_RT (82) SOL_MAX_RT Option (Max Solicit timeout
          value).";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.24";
      container sol-max-rt-option {
        description
          "OPTION_SOL_MAX_RT (82) SOL_MAX_RT option container.";
        leaf sol-max-rt-value {
          type dhc6:timer-seconds32;
          description
            "sol max rt value";
```

```
        }
      }
    }

    grouping inf-max-rt-option-group {
      description
        "OPTION_INF_MAX_RT (83) INF_MAX_RT Option (Max
          Information-request timeout value).";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.25";
      container inf-max-rt-option {
        description
          "OPTION_INF_MAX_RT (83) inf max rt option
          container.";
        leaf inf-max-rt-value {
          type dhc6:timer-seconds32;
          description
            "inf max rt value";
        }
      }
    }

    /*
     * Data Nodes
     */

    container dhcpv6-server {
      description
        "Configuration nodes for the DHCPv6 server.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 18.3";
      leaf enabled {
        type boolean;
        description
          "Enables the DHCP server function.";
      }
      leaf server-duid {
        type dhc6:duid;
        description
          "DUID of the server.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 11";
      }
      container vendor-config {
        description
          "This container provides a location for
          augmenting vendor or implementation specific
          configuration nodes.";
```

```
      }
      container option-sets {
        description
          "A server may allow different option sets
          to be configured for clients matching specific parameters
          such as topological location or client type. The
          'option-set' list is a set of options and their
          contents that will be returned to clients.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 21";
        list option-set {
          key option-set-id;
          description
            "YANG definitions for DHCPv6 options are
            contained in separate YANG modules and augmented to this
            container as required.";
          leaf option-set-id {
            type uint32;
            description
              "Option set identifier.";
          }
          leaf description {
            type string;
            description
              "An optional field for storing additional information
              relevant to the option set.";
          }
          uses preference-option-group;
          uses dhc6:auth-option-group;
          uses server-unicast-option-group;
          uses dhc6:status-code-option-group;
          uses dhc6:rapid-commit-option-group;
          uses dhc6:vendor-specific-information-option-group;
          uses reconfigure-message-option-group;
          uses dhc6:reconfigure-accept-option-group;
          uses info-refresh-time-option-group;
          uses sol-max-rt-option-group;
          uses inf-max-rt-option-group;
        }
      }

      container class-selector {
        description
          "DHCPv6 servers use a 'class-selector' function in order
          to identify and classify incoming client messages
          so that they can be given the correct configuration.
          The mechanisms used for implementing this function vary
          greatly between different implementations such it is not
```

```
          possible to include in this module. This container provides
          a location for server implementors to augment their own
          class-selector YANG.";
        }

        container network-ranges {
          description
            "This model is based on an address and parameter
            allocation hierarchy.  The top level is 'global' - which
            is defined as the container for all network-ranges. Under
            this are the individual network-ranges.";
          uses resource-config;
          list network-range {
            key id;
            description
              "Network-ranges are identified by the 'id' key.";
            leaf id {
              type uint32;
              mandatory true;
              description
                "Unique identifier for the network range.";
            }
            leaf description {
              type string;
              description
                "Description for the network range.";
            }
            leaf network-prefix {
              type inet:ipv6-prefix;
              mandatory true;
              description
                "Network prefix.";
            }
            uses resource-config;
            container address-pools {
              description
                "Configuration for the DHCPv6 server's
                address pools.";
              list address-pool {
                key pool-id;
                description
                  "List of address pools for allocation to clients,
                  distinguished by 'pool-id'.";
                leaf pool-id {
                  type uint32;
                  mandatory true;
                  description
                    "Unique identifier for the pool.";
```

```
        }
        leaf pool-prefix {
          type inet:ipv6-prefix;
          mandatory true;
          description
            "IPv6 prefix for the pool.";
        }
        leaf start-address {
          type inet:ipv6-address-no-zone;
          mandatory true;
          description
            "Starting IPv6 address for the pool.";
        }
        leaf end-address {
          type inet:ipv6-address-no-zone;
          mandatory true;
          description
            "Ending IPv6 address for the pool.";
        }
        leaf max-address-utilization {
          type dhc6:threshold;
          description
            "Maximum amount of the addresses in the
            pool which can be simultaneously allocated,
            calculated as a percentage of the available
            addresses (end-address minus start-address plus
            one).";
        }
        uses resource-config;
        container host-reservations {
          description
            "Configuration for host reservations from the
            address pool.";
          list host-reservation {
            key reserved-addr;
            description
              "List of host reservations.";
            leaf client-duid {
              type dhc6:duid;
              description
                "Client DUID for the reservation.";
            }
            leaf reserved-addr {
              type inet:ipv6-address;
              description
                "Reserved IPv6 address.";
            }
            uses resource-config;
```

```
        }
      }
      container active-leases {
        config false;
        description
          "Holds state related to active client
          leases.";
        leaf total-count {
          type uint64;
          mandatory true;
          description
            "The total number of addresses in the
            pool.";
        }
        leaf allocated-count {
          type uint64;
          mandatory true;
          description
            "The number of addresses or prefixes
            in the pool that are currently allocated.";
        }
        list active-lease {
          key leased-address;
          description
            "List of active address leases.";
          leaf leased-address {
            type inet:ipv6-address;
            description
              "Active address lease entry.";
          }
          uses lease-information;
        }
      }
    }
  }
  container prefix-pools {
    if-feature prefix-delegation;
    description
      "Configuration for the DHCPv6 server's prefix pools.";
    list prefix-pool {
      key pool-id;
      description
        "List of prefix pools for allocation to
        clients, distinguished by 'pool-id'.";
      leaf pool-id {
        type uint32;
        mandatory true;
        description
```

```
        "Unique identifier for the pool.";
      }
      leaf pool-prefix {
        type inet:ipv6-prefix;
        mandatory true;
        description
          "IPv6 prefix for the pool.";
      }
      leaf client-prefix-length {
        type uint8 {
          range "1 .. 128";
        }
        mandatory true;
        description
          "Length of the prefixes that will be delegated
          to clients.";
      }
      leaf max-pd-space-utilization {
        type dhc6:threshold;
        description
          "Maximum amount of the prefixes in the
          pool which can be simultaneously allocated,
          calculated as a percentage of the available
          prefixes, rounded up.";
      }
      uses resource-config;
      container host-reservations {
        description
          "Configuration for host reservations from the
          prefix pool.";
        list prefix-reservation {
          key reserved-prefix;
          description
            "Reserved prefix reservation.";
          leaf client-duid {
            type dhc6:duid;
            description
              "Client DUID for the reservation.";
          }
          leaf reserved-prefix {
            type inet:ipv6-prefix;
            description
              "Reserved IPv6 prefix";
          }
          leaf reserved-prefix-len {
            type uint8;
            description
              "Reserved IPv6 prefix length.";
```

```
                  }
                }
                uses resource-config;
              }
              container active-leases {
                config false;
                description
                  "Holds state related to active client prefix
                  leases.";
                leaf total-count {
                  type uint64;
                  mandatory true;
                  description
                    "The total number of prefixes in the pool.";
                }
                leaf allocated-count {
                  type uint64;
                  mandatory true;
                  description
                   "The number of prefixes in the pool that are
                   currently allocated.";
                }
                list active-lease {
                  key leased-prefix;
                  description
                    "List of active prefix leases.";
                  leaf leased-prefix {
                    type inet:ipv6-prefix;
                    description
                      "Active leased prefix entry.";
                  }
                  uses lease-information;
                }
              }
            }
          }
          uses message-stats;
        }
      }

      /*
       * Notifications
       */

      notification address-pool-utilization-threshold-exceeded {
        description
          "Notification sent when the address pool
```

```
        utilization exceeds the threshold configured in
        max-address-utilization.";
      leaf pool-id {
        type leafref {
          path "/dhcpv6-server/network-ranges/network-range/" +
            "address-pools/address-pool/pool-id";
        }
        mandatory true;
        description
          "Leafref to the address pool that the notification
          is being generated for.";
      }
      leaf total-pool-addresses {
        type uint64;
        mandatory true;
        description
          "Total number of addresses in the pool
          (end-address minus start-address plus one).";
      }
      leaf max-allocated-addresses {
        type uint64;
        mandatory true;
        description
          "Maximum number of addresses that can be simultaneously
          allocated from the pool. This value may be less than
          count of total addresses. Calculated as the
          max-address-utilization (percentage) of the
          total-pool-addresses, rounded up.";
      }
      leaf allocated-address-count {
        type uint64;
        mandatory true;
        description
          "Number of addresses allocated from the pool.";
      }
    }

    notification prefix-pool-utilization-threshold-exceeded {
      if-feature prefix-delegation;
      description
        "Notification sent when the prefix pool utilization
        exceeds the threshold configured in
        max-pd-space-utilization.";
      leaf pool-id {
        type leafref {
          path "/dhcpv6-server/network-ranges/network-range/" +
            "prefix-pools/prefix-pool/pool-id";
        }
```

```
        mandatory true;
        description
          "Unique identifier for the pool.";
      }
      leaf total-pool-prefixes {
        type uint64;
        mandatory true;
        description
          "Total number of prefixes in the pool.";
      }
      leaf max-allocated-prefixes {
        type uint64;
        mandatory true;
        description
          "Maximum number of prefixes that can be simultaneously
          allocated from the pool. This value may be less than
          count of total prefixes. Calculated as the
          max-prefix-utilization (percentage) of the
          total-pool-prefixes, rounded up.";
      }
      leaf allocated-prefixes-count {
        type uint64;
        mandatory true;
        description
          "Number of prefixes allocated from the pool.";
      }
    }

    notification invalid-client-detected {
      description
        "Notification sent when the server detects an
        invalid client.";
      leaf message-type {
        type enumeration {
          enum solicit {
            description
              "Solicit (1) message.";
          }
          enum request {
            description
              "Request (3) message.";
          }
          enum confirm {
            description
              "Confirm (4) message.";
          }
          enum renew {
            description
```

```
              "Renew (5) message.";
          }
          enum rebind {
            description
              "Rebind (6) message.";
          }
          enum release {
            description
              "Release (8) message.";
          }
          enum decline {
            description
              "Decline (9) message.";
          }
          enum info-request {
            description
              "Information request (11) message.";
          }
        }
        description
          "The message type received by the server that has caused
          the error.";
      }
      leaf duid {
        type dhc6:duid;
        description
          "Client DUID.";
      }
      leaf description {
        type string;
        description
          "Description of the event (e.g., and error code or log
          message).";
      }
    }

    notification decline-received {
      description
        "Notification sent when the server has received a
        Decline (9) message from a client.";
      leaf duid {
        type dhc6:duid;
        description
          "Client DUID.";
      }
      list declined-resources {
        description
          "List of declined addresses and/or prefixes.";
```

```
        choice resource-type {
          description
            "Type of resource that has been declined.";
          case declined-address {
            leaf address {
              type inet:ipv6-address;
              description
                "Address that has been declined.";
            }
          }
          case declined-prefix {
            leaf prefix {
              type inet:ipv6-prefix;
              description
                "Prefix that has been declined.";
            }
          }
        }
      }
    }

    notification non-success-code-sent {
      description
        "Notification sent when the server responded
        to a client with non-success status code.";
      leaf status-code {
        type uint16;
        mandatory true;
        description
          "Status code returned to the client.";
      }
      leaf duid {
        type dhc6:duid;
        description
          "Client DUID.";
      }
    }

    /*
     * RPCs
     */

    rpc delete-address-lease {
      nacm:default-deny-all;
      description
        "Deletes a client's active address lease from the
        server's lease database. Note this will not cause the address
        to be revoked from the client, and the lease may be refreshed
```

```
        or renewed by the client.";
      input {
        leaf lease-address-to-delete {
          type leafref {
            path "../../dhcpv6-server/network-ranges/network-range" +
              "/address-pools/address-pool/active-leases" +
              "/active-lease/leased-address";
          }
          mandatory true;
          description
            "IPv6 address of an active lease that will be
            deleted from the server.";
        }
      }
      output {
        leaf return-message {
          type string;
          description
            "Response message from the server.";
        }
      }
    }
    rpc delete-prefix-lease {
      nacm:default-deny-all;
      if-feature prefix-delegation;
      description
        "Deletes a client's active prefix lease from the
        server's lease database. Note, this will not cause the prefix
        to be revoked from the client, and the lease may be refreshed
        or renewed by the client.";
      input {
        leaf lease-prefix-to-delete {
          type leafref {
            path "../../dhcpv6-server/network-ranges/network-range" +
              "/prefix-pools/prefix-pool/active-leases/active-lease" +
              "/leased-prefix";
          }
          mandatory true;
          description
            "IPv6 prefix of an active lease that will be deleted
            from the server.";
        }
      }
      output {
        leaf return-message {
          type string;
          description
            "Response message from the server.";
```

```
        }
      }
    }
  }
  <CODE ENDS>
```

3.2. DHCPv6 Relay YANG Module

This module imports typedefs from [RFC6991], [RFC8343].

```
  <CODE BEGINS> file "ietf-dhcpv6-relay@2021-06-01.yang"

  module ietf-dhcpv6-relay {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay";
    prefix "dhc6-rly";

    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-yang-types {
      prefix yang;
      reference
        "RFC 6991: Common YANG Data Types";
    }

    import ietf-dhcpv6-common {
      prefix dhc6;
      reference
        "RFC XXXX: To be updated on publication";
    }

    import ietf-interfaces {
      prefix if;
      reference
        "RFC 8343: A YANG Data Model for Interface Management";
    }

    import ietf-netconf-acm {
      prefix nacm;
      reference
        "RFC 8341: Network Configuration Access Control Model";
    }

    organization
```

```
      "IETF DHC (Dynamic Host Configuration) Working Group";

    contact
      "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
       WG List:  <mailto:dhcwg@ietf.org>
       Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
       Author:   Linhui Sun <lh.sunlinh@gmail.com>
       Editor:   Ian Farrer <ian.farrer@telekom.de>
       Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
       Author:   Zihao He <hezihao9512@gmail.com>
       Author:   Michal Nowikowski <godfryd@isc.org>";

    description
      "This YANG module defines components necessary for the
      configuration and management of DHCPv6 relays.

      Copyright (c) 2021 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

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

    /*
     * Features
     */

    feature prefix-delegation {
      description
        "Enable if the relay functions as a delegating router for
        DHCPv6 prefix delegation.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 6.3";
    }
```

```
    /*
     * Groupings
     */

    grouping pd-lease-state {
      description
        "State data for the relay.";
      list pd-leases {
        key ia-pd-prefix;
        config false;
        description
          "Information about an active IA_PD prefix
          delegation.";
       leaf ia-pd-prefix {
          type inet:ipv6-prefix;
          description
            "Prefix that is delegated.";
        }
        leaf last-renew {
          type yang:date-and-time;
          description
            "Time of the last successful refresh or renew of the
            delegated prefix.";
        }
        leaf client-peer-address {
          type inet:ipv6-address;
          description
            "Peer-address of the leasing client.";
        }
        leaf client-duid {
          type dhc6:duid;
          description
            "DUID of the leasing client.";
        }
        leaf server-duid {
          type dhc6:duid;
          description
            "DUID of the delegating server.";
        }
      }
    }

    grouping message-statistics {
      description
        "Contains counters for the different DHCPv6
        message types.";
      leaf solicit-received-count {
        type uint32;
```

```
        config "false";
        description
          "Number of Solicit (1) messages received.";
      }
      leaf advertise-sent-count {
        type uint32;
        config "false";
        description
          "Number of Advertise (2) messages sent.";
      }
      leaf request-received-count {
        type uint32;
        config "false";
        description
          "Number of Request (3) messages received.";
      }
      leaf confirm-received-count {
        type uint32;
        config "false";
        description
          "Number of Confirm (4) messages received.";
      }
      leaf renew-received-count {
        type uint32;
        config "false";
        description
          "Number of Renew (5) messages received.";
      }
      leaf rebind-received-count {
        type uint32;
        config "false";
        description
          "Number of Rebind (6) messages received.";
      }
      leaf reply-sent-count {
        type uint32;
        config "false";
        description
          "Number of Reply (7) messages received.";
      }
      leaf release-received-count {
        type uint32;
        config "false";
        description
          "Number of Release (8) messages sent.";
      }
      leaf decline-received-count {
        type uint32;
```

```
        config "false";
        description
          "Number of Decline (9) messages sent.";
      }
      leaf reconfigure-sent-count {
        type uint32;
        config "false";
        description
          "Number of Reconfigure (10) messages sent.";
      }
      leaf information-request-received-count {
        type uint32;
        config "false";
        description
          "Number of Information-request (11) messages
          received.";
      }
      leaf unknown-message-received-count {
        type uint32;
        config "false";
        description
          "Number of messages of unknown type that have
          been received.";
      }
      leaf unknown-message-sent-count {
        type uint32;
        config "false";
        description
          "Number of messages of unknown type that have
          been sent.";
      }
      leaf discarded-message-count {
        type uint32;
        config "false";
        description
          "Number of messages that have been discarded
          for any reason.";
      }
    }

    grouping global-statistics {
      description
        "Global statistics for the device.";
      leaf relay-forward-sent-count {
        type uint32;
        config "false";
        description
          "Number of Relay-forward (12) messages sent.";
```

```
      }
      leaf relay-forward-received-count {
        type uint32;
        config "false";
        description
          "Number of Relay-forward (12) messages received.";
      }
      leaf relay-reply-received-count {
        type uint32;
        config "false";
        description
          "Number of Relay-reply (13) messages received.";
      }
      leaf relay-forward-unknown-sent-count {
        type uint32;
        config "false";
        description
          "Number of Relay-forward (12) messages containing
          a message of unknown type sent.";
      }
      leaf relay-forward-unknown-received-count {
        type uint32;
        config "false";
        description
          "Number of Relay-forward (12) messages containing
          a message of unknown type received.";
      }
      leaf discarded-message-count {
        type uint32;
        config "false";
        description
          "Number of messages that have been discarded
          for any reason.";
      }
    }

    grouping interface-id-option-group {
      description
        "OPTION_INTERFACE_ID (18) Interface-Id Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 21.18";
      container interface-id-option {
       description
         "OPTION_INTERFACE_ID (18) Interface-Id Option
         container.";
       leaf interface-id {
          type string;
          description
```

```
            "An opaque value of arbitrary length generated by the
            relay agent to identify one of the relay agent's
            interfaces.";
        }
      }
    }

    /*
     * Data Nodes
     */

    container dhcpv6-relay {
      description
        "This container contains the configuration data nodes
        for the relay.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 19";
     leaf enabled {
        type boolean;
        description
          "Globally enables the DHCP relay function.";
      }
      list relay-if {
        key if-name;
        description
          "List of interfaces configured for DHCPv6
          relaying.";
        leaf if-name {
          type if:interface-ref;
          description
            "interface-ref to the relay interface.";
        }
        leaf enabled {
          type boolean;
          description
            "Enables the DHCP relay function for this
            interface.";
        }
        leaf-list destination-address {
          type inet:ipv6-address;
          description
            "Each DHCPv6 relay agent may be configured with a list
            of destination addresses for relayed messages.
            The list may include unicast addresses, multicast
            addresses or other valid addresses.";
        }
        leaf link-address {
          type inet:ipv6-address;
```

```
          description
            "An address that may be used by the server to identify
            the link on which the client is located.";
        }
        container relay-options {
          description
            "Definitions for DHCPv6 options that can be sent
            by the relay are augmented to this location from other
            YANG modules as required.";
          uses dhc6:auth-option-group;
          uses dhc6:status-code-option-group;
          uses interface-id-option-group;
        }
        uses message-statistics;
        container prefix-delegation {
          if-feature prefix-delegation;
          presence "Enables prefix delegation for this interface.";
          description
            "Controls and holds state information for prefix
            delegation.";
          uses pd-lease-state;
        }
      }
      uses global-statistics;
    }

    /*
     * Notifications
     */

    notification relay-event {
      description
        "DHCPv6 relay event notifications.";
      container topology-change {
        description
          "Raised if the entry for an interface with DHCPv6
          related configuration or state is removed from
          if:interface-refs.";
        leaf relay-if-name {
          type leafref {
            path "/dhcpv6-relay/relay-if/if-name";
          }
          description
            "Name of the interface that has been removed.";
        }
        leaf last-ipv6-addr {
          type inet:ipv6-address;
          description
```

```
            "Last IPv6 address configured on the interface.";
        }
      }
    }

    /*
     * RPCs
     */

    rpc clear-prefix-entry {
      nacm:default-deny-all;
      if-feature prefix-delegation;
      description
        "Clears an entry for an active delegated prefix
        from the relay.";
      reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
        Section 4.4";
      input {
        leaf lease-prefix {
          type leafref {
            path "/dhcpv6-relay/relay-if/prefix-delegation" +
            "/pd-leases/ia-pd-prefix";
          }
          mandatory true;
          description
            "IPv6 prefix of an active lease entry that will
            be deleted from the relay.";
        }
      }
      output {
        leaf return-message {
          type string;
          description
            "Response message from the relay.";
        }
      }
    }
    rpc clear-client-prefixes {
      nacm:default-deny-all;
      if-feature prefix-delegation;
      description
        "Clears all active prefix entries for a single client.";
      reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
        Section 4.4";
     input {
        leaf client-duid {
          type dhc6:duid;
          mandatory true;
```

```
          description
            "DUID of the client.";
        }
      }
      output {
        leaf return-message {
          type string;
          description
            "Response message from the relay.";
        }
      }
    }
    rpc clear-interface-prefixes {
      nacm:default-deny-all;
      if-feature prefix-delegation;
      description
        "Clears all delegated prefix bindings from an
        interface on the relay.";
      reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
        Section 4.4";
      input {
        leaf interface {
          type leafref {
            path "../../dhcpv6-relay/relay-if/if-name";
          }
          mandatory true;
          description
            "Reference to the relay interface that will have all
            active prefix delegation bindings deleted.";
        }
      }
      output {
        leaf return-message {
          type string;
          description
            "Response message from the relay.";
        }
      }
    }
  }
  <CODE ENDS>
```

3.3. DHCPv6 Client YANG Module

This module imports typedefs from [RFC6991], [RFC8343].

```
   <CODE BEGINS> file "ietf-dhcpv6-client@2021-06-01.yang"

   module ietf-dhcpv6-client {
     yang-version 1.1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client";
     prefix "dhcpv6-clnt";

     import ietf-inet-types {
       prefix inet;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-yang-types {
       prefix yang;
       reference
         "RFC 6991: Common YANG Data Types";
     }

     import ietf-dhcpv6-common {
       prefix dhc6;
       reference
         "RFC XXXX: To be updated on publication";
     }

     import ietf-interfaces {
       prefix if;
       reference
         "RFC 8343: A YANG Data Model for Interface Management";
     }

     organization
       "IETF DHC (Dynamic Host Configuration) Working Group";

     contact
       "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
        WG List:  <mailto:dhcwg@ietf.org>
        Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
        Author:   Linhui Sun <lh.sunlinh@gmail.com>
        Editor:   Ian Farrer <ian.farrer@telekom.de>
        Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
        Author:   Zihao He <hezihao9512@gmail.com>
        Author:   Michal Nowikowski <godfryd@isc.org>";

     description
       "This YANG module defines components necessary for the
       configuration and management of DHCPv6 clients.
```

```
      The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
      'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
      'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
      are to be interpreted as described in BCP 14 (RFC 2119)
      (RFC 8174) when, and only when, they appear in all
      capitals, as shown here.

      Copyright (c) 2021 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

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

    /*
     * Groupings
     */

    grouping message-statistics {
      description
        "Counters for DHCPv6 messages.";
      leaf solicit-count {
        type uint32;
        config "false";
        description
          "Number of Solicit (1) messages sent.";
      }
      leaf advertise-count {
        type uint32;
        config "false";
        description
          "Number of Advertise (2) messages received.";
      }
      leaf request-count {
        type uint32;
        config "false";
```

```
        description
          "Number of Request (3) messages sent.";
      }
      leaf confirm-count {
        type uint32;
        config "false";
        description
          "Number of Confirm (4) messages sent.";
      }
      leaf renew-count {
        type uint32;
        config "false";
        description
          "Number of Renew (5) messages sent.";
      }
      leaf rebind-count {
        type uint32;
        config "false";
        description
          "Number of Rebind (6) messages sent.";
      }
      leaf reply-count {
        type uint32;
        config "false";
        description
          "Number of Reply (7) messages received.";
      }
      leaf release-count {
        type uint32;
        config "false";
        description
          "Number of Release (8) messages sent.";
      }
      leaf decline-count {
        type uint32;
        config "false";
        description
          "Number of Decline (9) messages sent.";
      }
      leaf reconfigure-count {
        type uint32;
        config "false";
        description
          "Number of Reconfigure (10) messages received.";
      }
      leaf information-request-count {
        type uint32;
        config "false";
```

```
        description
          "Number of Information-request (11) messages
          sent.";
      }
    }

    grouping option-request-option-group {
      description
        "OPTION_ORO (6) Option Request Option. A client MUST include
        an Option Request option in a Solicit, Request, Renew,
        Rebind, or Information-request message to inform the server
        about options the client wants the server to send to the
        client.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.7";
      container option-request-option {
        description
          "OPTION_ORO (6) Option Request Option container.";
        leaf-list oro-option {
          type uint16;
         description
           "List of options that the client is requesting,
           identified by option code";
        }
      }
    }

    grouping user-class-option-group {
      description
        "OPTION_USER_CLASS (15) User Class Option";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 21.15";
      container user-class-option {
        presence "Configures the option";
        description
          "OPTION_USER_CLASS (15) User Class Option
          container.";
        list user-class-data-instance {
          key user-class-data-id;
          min-elements 1;
          description
            "The user classes of which the client
            is a member.";
          leaf user-class-data-id {
            type uint8;
            description
              "User class data ID";
          }
```

```
          leaf user-class-data {
            type string;
            description
              "Opaque field representing a User Class
              of which the client is a member.";
          }
        }
      }
    }

    grouping vendor-class-option-group {
      description
        "OPTION_VENDOR_CLASS (16) Vendor Class Option";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 21.16";
      container vendor-class-option {
        description
          "OPTION_VENDOR_CLASS (16) Vendor Class Option
          container.";
        list vendor-class-option-instances {
          key enterprise-number;
          description
            "The vendor class option allows for multiple
            instances in a single message. Each list entry defines
            the contents of an instance of the option.";
          leaf enterprise-number {
            type uint32;
            description
              "The vendor's registered Enterprise Number as
              maintained by IANA.";
          }
          list vendor-class-data-element {
            key vendor-class-data-id;
            description
              "The vendor classes of which the client is a member.";
            leaf vendor-class-data-id {
              type uint8;
              description
                "Vendor class data ID";
            }
            leaf vendor-class-data {
              type string;
              description
                "Opaque field representing a vendor class of which
                the client is a member.";
            }
          }
        }
```

```
      }
    }

    /*
     * Data Nodes
     */

    container dhcpv6-client {
      description
        "DHCPv6 client configuration and state.";
      leaf enabled {
        type boolean;
        default true;
        description
          "Globally enables the DHCP client function.";
      }
      list client-if {
        key if-name;
        description
          "The list of interfaces for which the client will
          be requesting DHCPv6 configuration.";
        leaf if-name {
          type if:interface-ref;
          mandatory true;
          description
            "Reference to the interface entry that the requested
            configuration is relevant to.";
        }
        leaf enabled {
          type boolean;
          default true;
          description
            "Enables the DHCP client function for this interface.";
        }
        leaf duid {
          type dhc6:duid;
          description
            "Client DUID.";
          reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 11";
        }
        container client-configured-options {
          description
            "Definitions for DHCPv6 options that can be be sent by
            the client. Additional option definitions can be
            augmented to this location from other YANG modules as
            required.";
          uses option-request-option-group;
```

```
        uses dhc6:status-code-option-group;
        uses dhc6:rapid-commit-option-group;
        uses user-class-option-group;
        uses vendor-class-option-group;
        uses dhc6:vendor-specific-information-option-group;
        uses dhc6:reconfigure-accept-option-group;
      }
      list ia-na {
        key ia-id;
        description
          "Configuration relevant for an IA_NA (Identity Association
           for Non-temporary Addresses).";
        reference "RFC 8415: Dynamic Host Configuration Protocol
          for IPv6 (DHCPv6), Section 13.1";
        leaf ia-id {
          type uint32;
          description
            "A unique identifier for this IA_NA.";
          reference "RFC 8415: Dynamic Host Configuration Protocol
            for IPv6 (DHCPv6), Section 12";
        }
        container ia-na-options {
          description
            "An augmentation point for additional options
            that the client may send in the IA_NA-options field
            of OPTION_IA_NA.";
        }
        container lease-state {
          config "false";
          description
            "Information about the active IA_NA lease.";
          leaf ia-na-address {
            type inet:ipv6-address;
            description
              "Address that is currently leased.";
          }
          leaf preferred-lifetime {
            type dhc6:timer-seconds32;
            description
              "The preferred lifetime for the leased address
              expressed in seconds.";
          }
          leaf valid-lifetime {
            type dhc6:timer-seconds32;
            description
              "The valid lifetime for the leased address expressed
              in seconds.";
          }
```

```
            leaf lease-t1 {
              type dhc6:timer-seconds32;
              description
                "The time interval after which the client should
                contact the server from which the addresses in the
                IA_NA were obtained to extend the lifetimes of the
                addresses assigned to the IA_NA.";
            }
            leaf lease-t2 {
              type dhc6:timer-seconds32;
              description
                "The time interval after which the client should
                contact any available server to extend the lifetimes
                of the addresses assigned to the IA_NA.";
            }
            leaf allocation-time {
              type yang:date-and-time;
              description
                "Time and date that the address was first leased.";
            }
            leaf last-renew-rebind {
              type yang:date-and-time;
              description
                "Time of the last successful renew or rebind of the
                leased address.";
            }
            leaf server-duid {
              type dhc6:duid;
              description
                "DUID of the leasing server.";
            }
          }
        }
        list ia-ta {
          key ia-id;
          description
            "Configuration relevant for an IA_TA (Identity Association
            for Temporary Addresses).";
          reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 13.2";
          leaf ia-id {
            type uint32;
            description
              "The unique identifier for this IA_TA.";
            reference "RFC 8415: Dynamic Host Configuration Protocol
              for IPv6 (DHCPv6), Section 12";
          }
          container ia-ta-options {
```

```
            description
              "An augmentation point for additional options
              that the client may send in the IA_TA-options field
              of OPTION_IA_TA.";
          }
          container lease-state {
            config "false";
            description
              "Information about an active IA_TA lease.";
            leaf ia-ta-address {
              type inet:ipv6-address;
              description
                "Address that is currently leased.";
            }
            leaf preferred-lifetime {
              type dhc6:timer-seconds32;
              description
                "The preferred lifetime for the leased address
                expressed in seconds.";
            }
            leaf valid-lifetime {
              type dhc6:timer-seconds32;
              description
                "The valid lifetime for the leased address expressed
                in seconds.";
            }
            leaf allocation-time {
              type yang:date-and-time;
              description
                "Time and date that the address was first leased.";
            }
            leaf last-renew-rebind {
              type yang:date-and-time;
              description
                "Time of the last successful renew or rebind of the
                address.";
            }
            leaf server-duid {
              type dhc6:duid;
              description
                "DUID of the leasing server.";
            }
          }
        }
        list ia-pd {
          key ia-id;
          description
            "Configuration relevant for an IA_PD (Identity Association
```

```
        for Prefix Delegation).";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 13.3";
      leaf ia-id {
        type uint32;
        description
          "The unique identifier for this IA_PD.";
        reference "RFC 8415: Dynamic Host Configuration Protocol
          for IPv6 (DHCPv6), Section 12";
      }
      container ia-pd-options {
        description
          "An augmentation point for additional options that the
          client will send in the IA_PD-options field of
          OPTION_IA_TA.";
      }
      container lease-state {
        config "false";
        description
          "Information about an active IA_PD delegated prefix.";
        leaf ia-pd-prefix {
          type inet:ipv6-prefix;
          description
            "Delegated prefix that is currently leased.";
        }
        leaf preferred-lifetime {
          type dhc6:timer-seconds32;
          description
            "The preferred lifetime for the leased prefix expressed

            in units of seconds.";
        }
        leaf valid-lifetime {
          type dhc6:timer-seconds32;
          description
            "The valid lifetime for the leased prefix expressed
            in units of seconds.";
        }
        leaf lease-t1 {
          type dhc6:timer-seconds32;
          description
            "The time interval after which the client should
            contact the server from which the addresses in the
            IA_NA were obtained to extend the lifetimes of the
            addresses assigned to the IA_PD.";
        }
        leaf lease-t2 {
          type dhc6:timer-seconds32;
```

```
            description
              "The time interval after which the client should
              contact any available server to extend the lifetimes
              of the addresses assigned to the IA_PD.";
          }
          leaf allocation-time {
            type yang:date-and-time;
            description
              "Time and date that the prefix was first leased.";
          }
          leaf last-renew-rebind {
            type yang:date-and-time;
            description
              "Time of the last successful renew or rebind of the
              delegated prefix.";
          }
          leaf server-duid {
            type dhc6:duid;
            description
              "DUID of the delegating server.";
          }
        }
      }
      uses message-statistics;
    }
  }

  /*
   * Notifications
   */

  notification invalid-ia-address-detected {
    description
      "Notification sent when an address received
      in an identity association option is determined invalid.
      Possible conditions include a duplicate or otherwise illegal
      address.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6), Section 18.2.10.1";
    leaf ia-id {
      type uint32;
      mandatory true;
      description
        "IA-ID";
    }
    leaf ia-na-t1-timer {
      type uint32;
      description
```

```
          "The value of the T1 time field for non-temporary address
          allocations (OPTION_IA_NA).";
      }
      leaf ia-na-t2-timer {
        type uint32;
        description
          "The value of the preferred-lifetime field for non-temporary
          address allocations (OPTION_IA_NA).";
      }
      leaf invalid-address {
        type inet:ipv6-address;
        description
          "The IP address which has been detected to be invalid.";
      }
      leaf preferred-lifetime {
        type uint32;
        description
          "The value of the preferred-lifetime field in
          OPTION_IAADDR.";
      }
      leaf valid-lifetime {
        type uint32;
        description
          "The value of the valid-lifetime field in OPTION_IAADDR.";
      }
      leaf ia-options {
       type binary;
       description
         "A copy of the contents of the IAaddr-options field.";
      }
      leaf description {
        type string;
        description
          "Description of the invalid Identity Association (IA)
          detection error.";
      }
    }

    notification transmission-failed {
      description
        "Notification sent when the transmission or retransmission
        of a message fails.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 7.6";
      leaf failure-type {
        type enumeration {
          enum "solicit-timeout" {
            description
```

```
              "Max Solicit timeout value (SOL_MAX_RT) exceeded.";
          }
          enum "request-timeout" {
            description
              "Max Request timeout value (REQ_MAX_RT) exceeded.";
          }
          enum "request-retries-exceeded" {
            description
              "Max Request retry attempts (REC_MAX_RC) exceeded.";
          }
          enum "confirm-duration-exceeded" {
            description
              "Max Confirm duration (CNF_MAX_RD) exceeded.";
          }
          enum "renew-timeout" {
            description
              "Max Renew timeout value (REN_MAX_RT) exceeded.";
          }
          enum "rebind-timeout" {
            description
              "Max Rebind timeout value (REB_MAX_RT)
              exceeded.";
          }
          enum "info-request-timeout" {
            description
              "Max Information-request timeout value (INF_MAX_RT)
              exceeded.";
          }
          enum "release-retries-exceeded" {
            description
              "Max Release retry attempts (REL_MAX_RC) exceeded.";
          }
          enum "decline-retries-exceeded" {
            description
              "Max Decline retry attempts (DEC_MAX_RT) exceeded.";
          }
        }
        mandatory true;
        description
          "Description of the failure.";
      }
      leaf description {
        type string;
        description
          "Information related to the failure, such as number of
          retries and timer values.";
      }
    }
```

```
    notification unsuccessful-status-code {
      description
        "Notification sent when the client receives a message that
        includes an unsuccessful Status Code option.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 21.13";
      leaf status-code {
        type uint16;
        mandatory true;
        description
          "Unsuccessful status code received by a client.";
      }
      leaf server-duid {
        type dhc6:duid;
        mandatory true;
        description
          "DUID of the server sending the unsuccessful
          error code.";
      }
    }

    notification server-duid-changed {
      description
        "Notification sent when the client receives a lease
        from a server with different DUID to the one currently stored
        by the client, e.g., in response to a Rebind message.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 18.2.5";
      leaf new-server-duid {
        type dhc6:duid;
        mandatory true;
        description
          "DUID of the new server.";
      }
      leaf previous-server-duid {
        type dhc6:duid;
        mandatory true;
        description
          "DUID of the previous server.";
      }
      leaf lease-ia-na {
        type leafref {
          path "/dhcpv6-client/client-if/ia-na/ia-id";
        }
        description
          "Reference to the IA_NA lease.";
      }
      leaf lease-ia-ta {
```

```
        type leafref {
          path "/dhcpv6-client/client-if/ia-ta/ia-id";
        }
        description
          "Reference to the IA_TA lease.";
      }
      leaf lease-ia-pd {
        type leafref {
          path "/dhcpv6-client/client-if/ia-pd/ia-id";
        }
        description
          "Reference to the IA_PD lease.";
      }
    }
  }
  <CODE ENDS>
```

3.4. DHCPv6 Common YANG Module

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-common@2021-06-01.yang"

module ietf-dhcpv6-common {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-common";
  prefix "dhc6";

  organization
    "IETF DHC (Dynamic Host Configuration) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
     WG List:  <mailto:dhcwg@ietf.org>
     Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
     Author:   Linhui Sun <lh.sunlinh@gmail.com>
     Editor:   Ian Farrer <ian.farrer@telekom.de>
     Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
     Author:   Zihao He <hezihao9512@gmail.com>
     Author:   Michal Nowikowski <godfryd@isc.org>";

  description
    "This YANG module defines common components used for the
    configuration and management of DHCPv6.

    The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL',
    'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED',
    'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document
```

```
      are to be interpreted as described in BCP 14 (RFC 2119)
      (RFC 8174) when, and only when, they appear in all
      capitals, as shown here.

      Copyright (c) 2021 IETF Trust and the persons identified as
      authors of the code.  All rights reserved.

      Redistribution and use in source and binary forms, with or
      without modification, is permitted pursuant to, and subject
      to the license terms contained in, the Simplified BSD License
      set forth in Section 4.c of the IETF Trust's Legal Provisions
      Relating to IETF Documents
      (http://trustee.ietf.org/license-info).

      This version of this YANG module is part of RFC XXXX; see
      the RFC itself for full legal notices.";

    revision 2018-01-30 {
      description
        "Initial revision";
      reference "I-D: draft-ietf-dhc-dhcpv6-yang";
    }

    typedef threshold {
      type uint8 {
        range 1..100;
      }
      description
        "Threshold value in percent.";
    }

    typedef timer-seconds32 {
      type uint32;
      units "seconds";
      description
        "Timer value type, in seconds (32-bit range).";
    }

    typedef duid-base {
      type string {
        pattern '([0-9a-fA-F]{2}){3,130}';
      }
      description
        "Each DHCP server and client has a DUID (DHCP Unique
        Identifier). The DUID consists of a two-octet
        type field and an arbitrary length (1-128 octets) content
        field. The duid-base type provides the
        Currently, there are four defined types of DUIDs
```

```
        in RFC 8415 and RFC 6355 - DUID-LLT, DUID-EN, DUID-LL
        and DUID-UUID.  DUID-unstructured represents DUIDs which
        do not follow any of the defined formats.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11
        RFC 6355: Definition of the UUID-Based DHCPv6 Unique
        Identifier (DUID-UUID), Section 4";
    }

    typedef duid-llt {
      type duid-base {
        pattern '0001'
          + '[0-9a-fA-F]{12,}';
      }
      description
        "DUID type 1, based on Link-Layer Address Plus Time
        (DUID-LLT). Constructed with a 2-byte hardware type assigned
        by IANA,  4-bytes containing the time the DUID is generated
        (represented in seconds since midnight (UTC), January 1, 2000,
        modulo 2^32), and a link-layer address. The address is encoded
        without separator characters. For example:

        +------+----+----------+--------------+
        | 0001 | 06 | 28490058 | 00005E005300 |
        +------+----+----------+--------------+

        This example includes the 2-octet DUID type of 1 (0001), the
        hardware type is 06 (IEEE Hardware Types) the creation
        time is 0x284900580 (constructed as described above). Finally,
        the link-layer address is 00005E005300 (EUI-48 address
        00-00-5E-00-53-00)";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11.2
        IANA 'Hardware Types' registry.
          <https://www.iana.org/assignments/arp-parameters>";
    }

    typedef duid-en {
      type duid-base {
        pattern '0002'
          + '[0-9a-fA-F]{4,}';
      }
      description
        "DUID type 2, assigned by vendor based on Enterprise
        Number (DUID-EN). This DUID consists of the 4-octet vendor's
        registered Private Enterprise Number as maintained by IANA
        followed by a unique identifier assigned by the vendor. For
        example:
```

```
        +------+------+------------------+
        | 0002 | 0009 | 0CC084D303000912 |
        +------+------+------------------+

        This example includes the 2-octet DUID type of 2 (0002),
        4-octets for the Enterprise Number (0009), followed by 8-octets

        of identifier data (0x0CC084D303000912).";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11.3
        IANA 'Private Enterprise Numbers' registry.
          <https://www.iana.org/assignments/enterprise-numbers>";
    }

    typedef duid-ll {
      type duid-base {
        pattern '0003'
          + '([0-9a-fA-F]){4,}';
      }
      description
        "DUID type 3, based on Link-Layer Address (DUID-LL).
        Constructed with a 2-byte hardware type assigned
        by IANA, and a link-layer address. The address is encoded
        without separator characters. For example:

        +------+----+--------------+
        | 0003 | 06 | 00005E005300 |
        +------+----+--------------+

        This example includes the 2-octet DUID type of 3 (0003), the
        hardware type is 06 (IEEE Hardware Types), and the link-layer
        address is 00005E005300 (EUI-48 address 00-00-5E-00-53-00)";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11.4
        IANA 'Hardware Types' registry.
          <https://www.iana.org/assignments/arp-parameters>";
    }

    typedef duid-uuid {
      type duid-base {
        pattern '0004'
          + '[0-9a-fA-F]{32}';
      }
      description
        "DUID type 4, based on Universally Unique Identifier
        (DUID-UUID).  This type of DUID consists of 16 octets
        containing a 128-bit UUID. For example:
```

```
          +------+----------------------------------+
          | 0004 | 9f03b182705747e38a1e422910078642 |
          +------+----------------------------------+

          This example includes the 2-octet DUID type of 4 (0004), and
          the UUID 9f03b182-7057-47e3-8a1e-422910078642.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 11.5
          RFC 6355: Definition of the UUID-Based DHCPv6 Unique Identifier

          (DUID-UUID)";
      }

      typedef duid-unstructured {
        type duid-base {
          pattern '[0-9a-fA-F]{3}'
            + '[05-9a-fA-F]'
            + '([0-9a-fA-F])*';
        }
        description
          "Used for DUIDs following any other formats than DUID
          types 1-4. For example:

          +----------------------------------+
          | 7b6a164d325946539dc540fb539bc430 |
          +----------------------------------+

          Here, an arbitrary 16-octet value is used. The only constraint
          placed on this is that the first 2-octects are not 0001-0004
          to avoid collision with the other defined DUID types
          (duid-llt, duid-en, duid-ll, or duid-uuid).";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 11";
      }

      typedef duid {
        type union {
          type duid-llt;
          type duid-en;
          type duid-ll;
          type duid-uuid;
          type duid-unstructured;
        }
        description
          "Represents the DUID and is neutral to the DUID's construction
          format.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
          IPv6 (DHCPv6), Section 11";
```

```
    }

    /*
     * Groupings
     */

    grouping auth-option-group {
      description
      "OPTION_AUTH (11) Authentication Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 21.11
        IANA 'Dynamic Host Configuration Protocol (DHCP) Authentication

        Option Name Spaces' registry.
        <https://www.iana.org/assignments/auth-namespaces>";
      container auth-option {
        description
          "OPTION_AUTH (11) Authentication Option container.";
        leaf protocol {
          type uint8;
          description
            "The authentication protocol used by this Authentication
            option.";
        }
        leaf algorithm {
          type uint8;
          description
            "The algorithm used in the authentication protocol.";
        }
        leaf rdm {
          type uint8;
          description
            "The Replay Detection Method (RDM) used in this
            Authentication option.";
        }
        leaf replay-detection {
          type uint64;
          description
            "The replay detection information for the RDM.";
        }
        leaf auth-information {
          type string;
          description
            "The authentication information, as specified by the
            protocol and algorithm used in this Authentication
            option.";
        }
      }
```

```
    }

    grouping status-code-option-group {
      description
        "OPTION_STATUS_CODE (13) Status Code Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 21.13";
      container status-code-option {
        description
          "OPTION_STATUS_CODE (13) Status Code Option container.";
        leaf status-code {
          type uint16;
          description
            "The numeric code for the status encoded in this option.
            See the Status Codes registry at
            <https://www.iana.org/assignments/dhcpv6-parameters>
            for the current list of status codes.";
        }
        leaf status-message {
          type string;
          description
            "A UTF-8 encoded text string suitable for display to an
            end user. It MUST NOT be null-terminated.";
        }
      }
    }

    grouping rapid-commit-option-group {
      description
        "OPTION_RAPID_COMMIT (14) Rapid Commit Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 21.14";
      container rapid-commit-option {
        presence "Enable sending of this option";
        description
          "OPTION_RAPID_COMMIT (14) Rapid Commit Option container.";
      }
    }

    grouping vendor-specific-information-option-group {
      description
        "OPTION_VENDOR_OPTS (17) Vendor-specific Information
        Option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 21.17";
      container vendor-specific-information-options {
        description
          "OPTION_VENDOR_OPTS (17) Vendor-specific Information
```

```
          Option container.";
        list vendor-specific-information-option {
          key enterprise-number;
          description
            "The vendor-specific information option allows for
            multiple instances in a single message. Each list entry
            defines the contents of an instance of the option.";
          leaf enterprise-number {
            type uint32;
            description
              "The vendor's registered Enterprise Number, as
              maintained by IANA.";
            reference "IANA 'Private Enterprise Numbers' registry.
              <https://www.iana.org/assignments/enterprise-numbers>";
          }
          list vendor-option-data {
            key sub-option-code;
            description
              "Vendor options, interpreted by vendor-specific
              client/server functions.";
            leaf sub-option-code {
              type uint16;
              description
                "The code for the sub-option.";
            }
            leaf sub-option-data {
              type string {
                pattern '([0-9a-fA-F]{2}){0,}';
              }
              description
                "The data area for the sub-option.";
            }
          }
        }
      }
    }

    grouping reconfigure-accept-option-group {
      description
        "OPTION_RECONF_ACCEPT (20) Reconfigure Accept Option.
        A client uses the Reconfigure Accept option to announce to
        the server whether the client is willing to accept Reconfigure
        messages, and a server uses this option to tell the client
        whether or not to accept Reconfigure messages.  In the absence
        of this option, the default behavior is that the client is
        unwilling to accept Reconfigure messages.  The presence node
        is used to enable the option.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
```

```
        for IPv6 (DHCPv6), Section 21.20";
      container reconfigure-accept-option {
        presence "Enable sending of this option";
        description
          "OPTION_RECONF_ACCEPT (20)  Reconfigure Accept Option
          container.";
      }
    }
  }
  <CODE ENDS>
```

4. Security Considerations

The YANG modules defined in this document are designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

All data nodes defined in the YANG modules which can be created, modified, and deleted (i.e., config true, which is the default) are considered sensitive. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations.

As the RPCs for deleting/clearing active address and prefix entries in the server and relay modules are particularly sensitive, these use 'nacm:default-deny-all'.

An attacker who is able to access the DHCPv6 server can undertake various attacks, such as:

* Denial of service attacks, based on re-configuring messages to a rogue DHCPv6 server.

* Various attacks based on re-configuring the contents of DHCPv6 options. For example, changing the address of a the DNS server supplied in a DHCP option to point to a rogue server.

An attacker who is able to access the DHCPv6 relay can undertake various attacks, such as:

* Re-configuring the relay's destination address to send messages to a rogue DHCPv6 server.

* Deleting information about a client's delegated prefix, causing a denial of service attack as traffic will no longer be routed to the client.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. Therefore, it is important to control read access (e.g., only permitting get, get-config, or notifications) to these data nodes. These subtrees and data nodes can be misused to track the activity of a host:

* Information the server holds about clients with active leases: (dhc6-srv/network-ranges/network-range/address-pools/ address-pool/active-leases)

* Information the relay holds about clients with active leases: (dhc6-rly/relay-if/prefix-delegation/)

Security considerations related to DHCPv6 are discussed in [RFC8415].

Security considerations given in [RFC7950] are also applicable here.

5. IANA Considerations

This document requests IANA to register the following URIs in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-common
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

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

```
name:           ietf-dhcpv6-server
namespace:      urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server
prefix:         dhcpv6-server
reference:      RFC XXXX YANG Data Model for DHCPv6 Configuration

name:           ietf-dhcpv6-relay
namespace:      urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay
prefix:         dhcpv6-client
reference:      RFC XXXX YANG Data Model for DHCPv6 Configuration

name:           ietf-dhcpv6-client
namespace:      urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client
prefix:         dhcpv6-relay
reference:      RFC XXXX YANG Data Model for DHCPv6 Configuration

name:           ietf-dhcpv6-common
namespace:      urn:ietf:params:xml:ns:yang:ietf-dhcpv6-common
prefix:         dhcpv6-common
reference:      RFC XXXX YANG Data Model for DHCPv6 Configuration
```

6. Acknowledgments

The authors would like to thank Qi Sun, Lishan Li, Hao Wang, Tomek Mrugalski, Marcin Siodelski, Bernie Volz, Ted Lemon, Bing Liu, Tom Petch, and Acee Lindem for their valuable comments and contributions to this work.

7. Contributors

The following individuals are co-authors of this document:

Yong Cui
Tsinghua University
Beijing, 100084
P.R. China
Email: cuiyong@tsinghua.edu.cn

Linhui Sun
Tsinghua University
Beijing, 100084
P.R. China
Email: lh.sunlinh@gmail.com

Sladjana Zechlin
Deutsche Telekom AG
CTO-IPT, Landgrabenweg 151
53227, Bonn
Germany
Email: sladjana.zechlin@telekom.de

Zihao He
Tsinghua University
Beijing, 100084
P.R. China
Email: hezihao9512@gmail.com

Michal Nowikowski
Internet Systems Consortium
Gdansk
Poland
Email: godfryd@isc.org

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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

[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>.

[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>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <https://www.rfc-editor.org/info/rfc8415>.

[RFC8987] Farrer, I., Kottapalli, N., Hunek, M., and R. Patterson, "DHCPv6 Prefix Delegating Relay Requirements", RFC 8987, DOI 10.17487/RFC8987, February 2021, <https://www.rfc-editor.org/info/rfc8987>.

[IANA-HARDWARE-TYPES]
Internet Assigned Numbers Authority, "Hardware Types", <https://www.iana.org/assignments/arp-parameters>.

[IANA-PEN] Internet Assigned Numbers Authority, "Private Enterprise Numbers", <https://www.iana.org/assignments/enterprise-numbers>.

[IANA-AUTH]
Internet Assigned Numbers Authority, "Dynamic Host Configuration Protocol (DHCP) Authentication Option Name Spaces", <https://www.iana.org/assignments/auth-namespaces>.

[IANA-STATUS]
Internet Assigned Numbers Authority, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Status Codes", <https://www.iana.org/assignments/dhcpv6-parameters>.

8.2. Informative References

[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>.

Appendix A. Data Tree Examples

This section contains XML examples of data trees for the different DHCPv6 elements.

A.1. DHCPv6 Server Configuration Example

The following example shows a basic configuration for a server. The configuration defines:

* Enabling the DHCP server function

* The server's DUID

* An option set (id=1) with configuration for the Solicit Max Retry Timeout (SOL_MAX_RT (82)) option.

* A single network range (2001:db8::/32)

* A single address pool, with start and end addresses, relevant lease timers and an option-set-id of "1" referencing the option set configured above.

```
<dhcpv6-server
    xmlns="urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server">
  <enabled>true</enabled>
  <server-duid>000200090CC084D303000912</server-duid>
  <vendor-config/>
  <option-sets>
    <option-set>
      <option-set-id>1</option-set-id>
      <description>Example DHCP option set</description>
      <sol-max-rt-option>
        <sol-max-rt-value>3600</sol-max-rt-value>
      </sol-max-rt-option>
    </option-set>
  </option-sets>
  <class-selector/>
  <network-ranges>
    <valid-lifetime>54000</valid-lifetime>
    <renew-time>7200</renew-time>
    <rebind-time>32400</rebind-time>
    <preferred-lifetime>43200</preferred-lifetime>
    <network-range>
      <id>1</id>
      <description>example-network-range</description>
      <network-prefix>2001:db8::/32</network-prefix>
      <option-set-id>1</option-set-id>
      <address-pools>
        <address-pool>
          <pool-id>1</pool-id>
          <pool-prefix>2001:db8:1:1::/64</pool-prefix>
          <start-address>2001:db8:1:1::1000</start-address>
          <end-address>2001:db8:1:1::2000</end-address>
          <max-address-utilization>50</max-address-utilization>
          <option-set-id>1</option-set-id>
        </address-pool>
      </address-pools>
    </network-range>
  </network-ranges>
</dhcpv6-server>
```

Figure 4: Basic Server Configuration Example XML

The following example shows a static host reservation within an address pool. The host's lease timers are configured to be longer than hosts from the pool with dynamically assigned addresses.

```
        <address-pools>
          <address-pool>
            <pool-id>1</pool-id>
            <pool-prefix>2001:db8:1:1::/64</pool-prefix>
            <start-address>2001:db8:1:1::1000</start-address>
            <end-address>2001:db8:1:1::2000</end-address>
            <max-address-utilization>50</max-address-utilization>
            <option-set-id>1</option-set-id>
            <host-reservations>
              <host-reservation>
                <reserved-addr>2001:db8:1:1::1001</reserved-addr>
                <client-duid>00052001db81</client-duid>
                <option-set-id>1</option-set-id>
                <valid-lifetime>604800</valid-lifetime>
                <renew-time>86400</renew-time>
                <rebind-time>172800</rebind-time>
                <preferred-lifetime>345600</preferred-lifetime>
              </host-reservation>
            </host-reservations>
          </address-pool>
        </address-pools>
```

Figure 5: Host Reservation Configuration Example XML

The following example shows configuration for a network range and pool to be used for delegating prefixes to clients. In this example, each client will receive a /56 prefix.

The 'max-pd-space-utiliation' is set to 80 so that a 'prefix-pool-utilization-threshold-exceeded' notification will be raised.

```
      <network-ranges>
        <network-range>
          <id>1</id>
          <description>prefix-pool-example</description>
          <network-prefix>2001:db8::/32</network-prefix>
          <prefix-pools>
            <valid-lifetime>54000</valid-lifetime>
            <renew-time>7200</renew-time>
            <rebind-time>32400</rebind-time>
            <preferred-lifetime>43200</preferred-lifetime>
            <prefix-pool>
              <pool-id>0</pool-id>
              <option-set-id>1</option-set-id>
              <pool-prefix>2001:db8:1::/48</pool-prefix>
              <client-prefix-length>56</client-prefix-length>
              <max-pd-space-utilization>80</max-pd-space-utilization>
            </prefix-pool>
          </prefix-pools>
        </network-range>
      </network-ranges>
```

Figure 6: Prefix Delegation Configuration Example XML

The next example shows the configuration of a set of options that may be returned to clients, depending on the contents of a received DHCP request message. The option set ID is '1', which will referenced by other places in the configuration (e.g., address pool configuration) as the available options for clients that request them.

The example contains 2 instances of the Vendor Specific Information Option (OPTION_VENDOR_OPTS (17)) to show how options which allow for multiple instances are configured.

```
<option-sets>
  <option-set>
    <option-set-id>1</option-set-id>
    <description>Example DHCP option set</description>
    <vendor-specific-information-options>
      <vendor-specific-information-option>
        <enterprise-number>9</enterprise-number>
        <vendor-option-data>
          <sub-option-code>01</sub-option-code>
          <sub-option-data>1234abcd</sub-option-data>
        </vendor-option-data>
        <vendor-option-data>
          <sub-option-code>02</sub-option-code>
          <sub-option-data>abcd1234</sub-option-data>
        </vendor-option-data>
      </vendor-specific-information-option>
      <vendor-specific-information-option>
        <enterprise-number>10</enterprise-number>
        <vendor-option-data>
          <sub-option-code>01</sub-option-code>
          <sub-option-data>4321dcba</sub-option-data>
        </vendor-option-data>
        <vendor-option-data>
          <sub-option-code>02</sub-option-code>
          <sub-option-data>dcba4321</sub-option-data>
        </vendor-option-data>
      </vendor-specific-information-option>
    </vendor-specific-information-options>
    <sol-max-rt-option>
      <sol-max-rt-value>3600</sol-max-rt-value>
    </sol-max-rt-option>
  </option-set>
</option-sets>
```

Figure 7: Option Set Configuration Example XML

A.2. DHCPv6 Relay Configuration Example

The following example shows a basic configuration for a single DHCP relay interface and its interaction with the ietf-interfaces module. The configuration defines:

* Enabling the DHCP relay function globally and for the relevant interface.

* Referencing the interface that the relay configuration is relevant for via an inteface-ref to the ietf-interfaces module.

* Defining two destination addresses that incoming DHCP messages will be relayed to.

* Configures the link-address value that will be sent in the relay-forward message.

* Configuring a value for the Interface ID Option (OPTION_INTERFACE_ID (18)), which will be included in the relay forward message.

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <description>DHCPv6 Relay Interface</description>
    <enabled>true</enabled>
  </interface>
</interfaces>
<dhcpv6-relay xmlns="urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay">
  <enabled>true</enabled>
  <relay-if>
    <if-name>eth0</if-name>
    <enabled>true</enabled>
    <destination-address>2001:db8:2::1</destination-address>
    <destination-address>2001:db8:2::2</destination-address>
    <link-address>2001:db8:3::1</link-address>
    <relay-options>
      <interface-id-option>
        <interface-id>EXAMPLE-INTERFACE-ID</interface-id>
      </interface-id-option>
    </relay-options>
  </relay-if>
</dhcpv6-relay>
```

Figure 8: Basic Relay Configuration Example XML

A.3. DHCPv6 Client Configuration Examples

The following example shows a basic configuration for a DHCP client and its interaction with the ietf-interfaces module. The configuration defines:

* Enabling the DHCP relay function globally and for the relevant interface.

* References the interface that the client configuration is relevant for via an inteface-ref to the ietf-interfaces module.

* Sets the client's DUID.

* Configures a list of option codes that will be requested by the client in its Option Request Option (OPTION_ORO (5)).

* Configures a single instance of the Vendor-specific Information Option (OPTION_VENDOR_OPTS (17)) with a single sub-option data item.

* Requests a non-temporary IPv6 address (IA_NA) with an identity association interface identifier of 00000001.

* Requests an IPv6 delegated prefix address (IA_PD) with an identity association interface identifier of 00000002.

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <description>DHCPv6 Relay Interface</description>
    <enabled>true</enabled>
  </interface>
</interfaces>
<dhcpv6-client xmlns="urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client"
>
  <enabled>true</enabled>
  <client-if>
    <if-name>eth0</if-name>
    <enabled>true</enabled>
    <duid>000200090CC084D303000913</duid>
    <client-configured-options>
      <option-request-option>
        <oro-option>17</oro-option>
        <oro-option>23</oro-option>
        <oro-option>24</oro-option>
        <oro-option>82</oro-option>
      </option-request-option>
      <vendor-specific-information-options>
        <vendor-specific-information-option>
          <enterprise-number>9</enterprise-number>
          <vendor-option-data>
            <sub-option-code>01</sub-option-code>
            <sub-option-data>abcd1234</sub-option-data>
          </vendor-option-data>
        </vendor-specific-information-option>
      </vendor-specific-information-options>
    </client-configured-options>
    <ia-na>
      <ia-id>00000001</ia-id>
    </ia-na>
    <ia-pd>
      <ia-id>00000002</ia-id>
    </ia-pd>
  </client-if>
</dhcpv6-client>
```

Figure 9: Basic Server Configuration Example XML

Appendix B. Example of Augmenting Additional DHCPv6 Option Definitions

The following section provides a example of how the DHCPv6 option definitions can be extended to include additional options. It is expected that additional specification documents will be published for this in the future.

The example defines YANG models for OPTION_SIP_SERVER_D (21) and OPTION_SIP_SERVER_D (22) defined in [RFC3319]. The module is constructed as follows:

* The module is named using a meaningful, shortened version of the document name in which the DHCP option format is specified.

* A separate grouping is used to define each option.

* The name of the option is taken from the registered IANA name for the option, with an '-option' suffix added.

* The description field is taken from the relevant option code name and number.

* The reference section is the number and name of the RFC in which the DHCPv6 option is defined.

* The remaining fields match the fields in the DHCP option. They are in the same order as defined in the DHCP option. Where-ever possible, the format that is defined for the DHCP field should be matched by the relevant YANG type.

* Fields which can have multiple entries or instances are defined using list or leaf-list nodes.

Below the groupings for option definitions, augment statements are used to add the option definitions for use in the relevant DHCP element's module (server, relay and/or client).

```
  module example-dhcpv6-opt-sip-serv {
    yang-version 1.1;
    namespace "https://example.com/ns/" +
      "example-dhcpv6-opt-sip-serv";
    prefix "sip-srv";

    import ietf-inet-types {
      prefix inet;
    }

    import ietf-dhcpv6-server {
```

```
  prefix dhc6-srv;
}

organization
  "IETF DHC (Dynamic Host Configuration) Working Group";

contact
  "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
   WG List:  <mailto:dhcwg@ietf.org>
   Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
   Author:   Linhui Sun <lh.sunlinh@gmail.com>
   Editor:   Ian Farrer <ian.farrer@telekom.de>
   Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
   Author:   Zihao He <hezihao9512@gmail.com>
   Author:   Michal Nowikowski <godfryd@isc.org>";

description
  "This YANG module contains DHCPv6 options defined in RFC 8415
  that can be used by DHCPv6 servers.

  Copyright (c) 2021 IETF Trust and the persons identified as
  authors of the code.  All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject
  to the license terms contained in, the Simplified BSD License
  set forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (http://trustee.ietf.org/license-info).

  This version of this YANG module is part of RFC XXXX; see
  the RFC itself for full legal notices.";

revision 2019-10-18 {
  description
    "Initial version.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

/*
 * Groupings
 */

grouping sip-server-domain-name-list-option-group {
  description
    "OPTION_SIP_SERVER_D (21) SIP Servers Domain-Name List";
  reference "RFC 3319: Dynamic Host Configuration Protocol
    (DHCPv6) Options for Session Initiation Protocol (SIP)
```

```
          Servers";
        container sip-server-domain-name-list-option {
         description
           "OPTION_SIP_SERVER_D (21) SIP Servers Domain Name List
           container.";
         list sip-server {
            key sip-serv-id;
            description
              "SIP server information.";
            leaf sip-serv-id {
              type uint8;
              description
                "SIP server list identifier identifier.";
            }
            leaf sip-serv-domain-name {
              type inet:domain-name;
              description
                "SIP server domain name.";
            }
          }
        }
      }

      grouping sip-server-address-list-option-group {
        description
          "OPTION_SIP_SERVER_A (22) SIP Servers IPv6 Address List";
        reference "RFC 3319: Dynamic Host Configuration Protocol
          (DHCPv6) Options for Session Initiation Protocol (SIP)
          Servers";
        container sip-server-address-list-option {
          description
            "OPTION_SIP_SERVER_A (22) SIP Servers IPv6 Address List
            container.";
         list sip-server {
            key sip-serv-id;
            description
              "SIP server information.";
            leaf sip-serv-id {
              type uint8;
              description
                "SIP server list entry identifier.";
            }
            leaf sip-serv-addr {
              type inet:ipv6-address;
              description
                "SIP server IPv6 address.";
            }
          }
```

```
        }
      }

      /*
       * Augmentations
       */

      augment "/dhc6-srv:dhcpv6-server/dhc6-srv:option-sets/" +
        "dhc6-srv:option-set" {
        description
          "Augment the option definition groupings to the server
          module.";
        uses sip-server-domain-name-list-option-group;
        uses sip-server-address-list-option-group;
      }
    }
```

The correct location to augment the new option definition(s) will vary according to the specific rules defined for the use of that specific option. For example, for options which will be augmented into the ietf-dhcpv6-server module, in many cases, these will be augmented to:

'/dhc6-srv:dhc6-srv/dhc6-srv:option-sets/dhc6-srv:option-set'

So that they can be defined within option sets. However, there are some options which are only applicable for specific deployment scenarios and in these cases it may be more logical to augment the option group to a location relevant for the option.

One example for this could be OPTION_PD_EXCLUDE (67). This option is only relevant in combination with a delegated prefix which contains a specific prefix. In this case, the following location for the augmentation may be more suitable:

'/dhc6-srv:dhc6-srv/dhc6-srv:network-ranges/dhc6-srv:network-range/
dhc6-srv:prefix-pools/dhc6-srv:prefix-pool"

Appendix C. Example Vendor Specific Server Configuration Module

This section shows how to extend the server YANG module defined in this document with vendor specific configuration nodes, e.g., configuring access to a lease storage database.

The example module defines additional server attributes such as name and description. Storage for leases is configured using a lease-storage container. It allows storing leases in one of three options: memory (memfile), MySQL and PosgreSQL. For each case, the necessary configuration parameters are provided.

At the end there is an augment statement which adds the vendor specific configuration defined in "dhcpv6-server-config:config" under the "/dhcpv6-server:config/dhcpv6-server:vendor-config" mount point.

```
module example-dhcpv6-server-conf {
  yang-version 1.1;
  namespace "https://example.com/ns/" +
    "example-dhcpv6-server-conf";
  prefix "dhc6-srv-conf";

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-dhcpv6-server {
    prefix dhc6-srv;
  }

  organization
    "IETF DHC (Dynamic Host Configuration) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
     WG List:  <mailto:dhcwg@ietf.org>
     Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
     Author:   Linhui Sun <lh.sunlinh@gmail.com>
     Editor:   Ian Farrer <ian.farrer@telekom.de>
     Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
     Author:   Zihao He <hezihao9512@gmail.com>
     Author:   Michal Nowikowski <godfryd@isc.org>";

  description
    "This YANG module defines components for the configuration and
    management of vendor/implementation specific DHCPv6 server
    functionality. As this functionality varies greatly between
    different implementations, the module is provided as an example
    only.
```

```
  revision 2019-06-04 {
    description
      "Initial revision.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
  }

  /*
   * Groupings
   */

  grouping config {
    description
      "Parameters necessary for the configuration of a DHCPv6
      server";
    container serv-attributes {
      description
        "Contains basic attributes necessary for running a DHCPv6
        server.";
      leaf name {
        type string;
        description
          "Name of the DHCPv6 server.";
      }
      leaf description {
        type string;
        description
          "Description of the DHCPv6 server.";
      }
      leaf ipv6-listen-port {
        type uint16;
        default 547;
        description
          "UDP port that the server will listen on.";
      }
      choice listening-interfaces {
```

```
          default all-interfaces;
          description
            "Configures which interface or addresses the server will
            listen for incoming messages on.";
          case all-interfaces {
            container all-interfaces {
            presence true;
            description
              "Configures the server to listen for incoming messages
              on all IPv6 addresses (unicast and multicast) on all of
              its network interfaces.";
            }
          }
          case interface-list {
            leaf-list interfaces {
              type if:interface-ref;
              description
                "List of interfaces on which the server will listen for

                incoming messages. Messages addressed to any
                valid IPv6 address (unicast and multicast) will be
                received.";
            }
          }
          case address-list {
            leaf-list address-list {
              type inet:ipv6-address;
              description
                "List of IPv6 address(es) on which the server will
                listen for incoming DHCPv6 messages.";
            }
          }
        }
        leaf-list interfaces-config {
          type if:interface-ref;
          default "if:interfaces/if:interface/if:name";
          description
            "A leaf list of interfaces on which the server should
            listen.";
        }
        container lease-storage {
          description
            "Configures how the server will stores leases.";
          choice storage-type {
            description
              "The type storage that will be used for lease
              information.";
            case memfile {
```

```
            description
              "Configuration for storing leases information in a
              Comma-Separated Value (CSV) file.";
            leaf memfile-name {
              type string;
              description
                "Specifies the absolute location of the lease file.
                The format of the string follow the semantics of
                the relevant operating system.";
            }
            leaf memfile-lfc-interval {
              type uint64;
              description
                "Specifies the interval in seconds, at which the
                server will perform a lease file cleanup (LFC).";
            }
          }
          case mysql {
            leaf mysql-name {
              type string;
              description
                "Name of the database.";
            }
            choice mysql-host {
              description
                "Define host or address for MySQL server.";
              case mysql-server-hostname {
                leaf mysql-hostname {
                  type inet:domain-name;
                  default "localhost";
                  description
                    "If the database is located on a different
                    system to the DHCPv6 server, the domain name can
                    be specified.";
                }
              }
              case mysql-server-address {
                leaf mysql-address {
                  type inet:ip-address;
                  default "::";
                  description
                    "Configure the location of the database using
                    an IP (v6 or v6) literal address";
                }
              }
            }
            leaf mysql-username {
              type string;
```

```
            description
              "User name of the account under which the server
              will access the database.";
          }
          leaf mysql-password {
            type string;
            description
              "Password of the account under which the server
              will access the database.";
          }
          leaf mysql-port {
            type inet:port-number;
            default 5432;
            description
              "If the database is located on a different system,
              the port number may be specified.";
          }
          leaf mysql-lfc-interval {
            type uint64;
            description
              "Specifies the interval in seconds, at which the
              server will perform a lease file cleanup (LFC).";
          }
          leaf mysql-connect-timeout {
            type uint64;
            description
              "Defines the timeout interval for connecting to the
              database. A longer interval can be specified if the
              database is remote.";
          }
        }
        case postgresql {
          choice postgresql-host {
            description
              "Define host or address for postgresql server.";
            case postgresql-server-hostname {
              leaf postgresql-hostname {
                type inet:domain-name;
                default "localhost";
                description
                  "If the database is located on a different system

                  to the DHCPv6 server, the domain name can be
                  specified.";
              }
            }
            case postgresql-server-address {
              leaf postgresql-address {
```

```
                type inet:ip-address;
                default "::";
                description
                  "Configure the location of the database using
                  an IP (v6 or v6) literal address";
              }
            }
          }
          leaf postgresql-username {
            type string;
            description
              "User name of the account under which the server
              will access the database";
          }
          leaf postgresql-password {
            type string;
            description
              "Password of the account under which the server
              will access the database";
          }
          leaf postgresql-port {
            type inet:port-number;
            default 5432;
            description
              "If the database is located on a different system,
              the port number may be specified";
          }
          leaf postgresql-lfc-interval {
            type uint64;
            description
              "Specifies the interval in seconds, at which the
              server will perform a lease file cleanup (LFC)";
          }
          leaf postgresql-connect-timeout {
            type uint64;
            description
              "Defines the timeout interval for connecting to the
              database. A longer interval can be specified if the
              database is remote.";
          }
        }
      }
    }
  }
}

/*
 * Augmentations
```

```
     */

    augment "/dhc6-srv:dhcpv6-server/dhc6-srv:vendor-config" {
      description
        "Augment the server specific YANG to the ietf-dhcpv6-server
        module.";
      uses config;
    }
  }
```

Appendix D. Example definition of class-selector configuration

The module "ietf-example-dhcpv6-class-selector" provides an example of how vendor-specific class selection configuration can be modelled and integrated with the "ietf-dhcpv6-server" module defined in this document.

The example module defines "client-class-names" with associated matching rules. A client can be classified based on "client-id", "interface-id" (ingress interface of the client's messages), packet's source or destination address, relay link address, relay link interface-id and more. Actually, there are endless methods for classifying clients. So this standard does not try to provide full specification for class selection, it only shows an example how it could be defined.

At the end of the example augment statements are used to add the defined class selector rules into the overall DHCPv6 addressing hierarchy. This is done in two main parts:

* The augmented class-selector configuration in the main DHCPv6 Server configuration.

* client-class leafrefs augmented to "network-range", "address-pool" and "pd-pool", pointing to the "client-class-name" that is required.

The mechanism is as follows: class is associated to client based on rules and then client is allowed to get address(es)/prefix(es) from given network-range/pool if the class name matches.

```
module example-dhcpv6-class-select {
  yang-version 1.1;
  namespace "https://example.com/ns/" +
    "example-dhcpv6-class-select";
  prefix "dhc6-class-sel";

  import ietf-inet-types {
```

```
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-dhcpv6-common {
    prefix dhc6;
  }

  import ietf-dhcpv6-server {
    prefix dhc6-srv;
  }

  organization
    "IETF DHC (Dynamic Host Configuration) Working Group";

  contact
    "WG Web:   <http://datatracker.ietf.org/wg/dhc/>
     WG List:  <mailto:dhcwg@ietf.org>
     Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
     Author:   Linhui Sun <lh.sunlinh@gmail.com>
     Editor:   Ian Farrer <ian.farrer@telekom.de>
     Author:   Sladjana Zeichlin <sladjana.zechlin@telekom.de>
     Author:   Zihao He <hezihao9512@gmail.com>
     Author:   Michal Nowikowski <godfryd@isc.org>";

  description
    "This YANG module defines components for the definition and
    configuration of the client class selector function for a
    DHCPv6 server.  As this functionality varies greatly between
    different implementations, the module provided as an example
    only.

    Copyright (c) 2021 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
    set forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (http://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX; see
    the RFC itself for full legal notices.";
```

```
    revision 2019-06-13 {
      description
        "Initial version.";
      reference "I-D: draft-ietf-dhc-dhcpv6-yang";
    }

    /*
     * Groupings
     */

    grouping client-class-id {
      description
        "Definitions of client message classification for
        authorization and assignment purposes.";
      leaf client-class-name {
        type string;
        description
          "Unique Identifier for client class identification list
          entries.";
      }
      choice id-type {
        mandatory true;
        description
          "Definitions for different client identifier types.";
        case client-id-id {
          leaf client-id {
            type string;
            mandatory true;
            description
              "String literal client identifier.";
          }
          description
            "Client class selection based on a string literal client
            identifier.";
        }
        case received-interface-id {
          description
            "Client class selection based on the incoming interface
            of the DHCPv6 message.";
          leaf received-interface {
            type if:interface-ref;
            description
              "Reference to the interface entry for the incoming
              DHCPv6 message.";
          }
        }
        case packet-source-address-id {
          description
```

```
            "Client class selection based on the source address of
            the DHCPv6 message.";
          leaf packet-source-address {
            type inet:ipv6-address;
            mandatory true;
            description
              "Source address of the DHCPv6 message.";
          }
        }
        case packet-destination-address-id {
          description
            "Client class selection based on the destination address
            of the DHCPv6 message.";
          leaf packet-destination-address {
            type inet:ipv6-address;
            mandatory true;
            description
              "Destination address of the DHCPv6 message.";
          }
        }
        case relay-link-address-id {
          description
            "Client class selection based on the prefix of the
            link-address field in the relay agent message header.";
          leaf relay-link-address {
            type inet:ipv6-prefix;
            mandatory true;
            description
              "Prefix of the link-address field in the relay agent
              message header.";
          }
        }
        case relay-peer-address-id {
          description
            "Client class selection based on the value of the
            peer-address field in the relay agent message header.";
          leaf relay-peer-address {
            type inet:ipv6-prefix;
            mandatory true;
            description
              "Prefix of the peer-address field in the relay agent
              message header.";
          }
        }
        case relay-interface-id {
          description
            "Client class selection based on the incoming
            interface-id option.";
```

```
            leaf relay-interface {
              type string;
              description
                "Reference to the interface entry for the incoming
                DHCPv6 message.";
            }
          }
          case user-class-option-id {
            description
              "Client class selection based on the value of the
              OPTION_USER_CLASS(15) and its user-class-data field.";
            leaf user-class-data {
              type string;
              mandatory true;
              description
                "Value of the enterprise-number field.";
            }
          }
          case vendor-class-present-id {
            description
              "Client class selection based on the presence of
              OPTION_VENDOR_CLASS(16) in the received message.";
            leaf vendor-class-present {
              type boolean;
              mandatory true;
              description
                "Presence of OPTION_VENDOR_CLASS(16) in the received
                message.";
            }
          }
          case vendor-class-option-enterprise-number-id {
            description
              "Client class selection based on the value of the
              enterprise-number field in OPTION_VENDOR_CLASS(16).";
            leaf vendor-class-option-enterprise-number {
              type uint32;
              mandatory true;
              description
                "Value of the enterprise-number field.";
            }
          }
          case vendor-class-option-data-id {
            description
              "Client class selection based on the value of a data
              field within a vendor-class-data entry for a matching
              enterprise-number field in OPTION_VENDOR_CLASS(16).";
            container vendor-class-option-data {
              description
```

```
            "Vendor class option data container.";
          leaf vendor-class-option-enterprise-number {
            type uint32;
            mandatory true;
            description
              "Value of the enterprise-number field for matching
              the data contents.";
          }
          leaf vendor-class-data {
            type string;
            mandatory true;
            description
              "Vendor class data to match.";
          }
        }
      }
      case remote-id {
        description
          "Client class selection based on the value of Remote-ID.";
        container remote-id {
          description
            "Remote-id client class selector container.";
          leaf vendor-class-option-enterprise-number {
            type uint32;
            mandatory true;
            description
              "Value of the enterprise-number field for matching the
              data contents.";
          }
          leaf remote-id {
            type string;
            mandatory true;
            description
              "Remote-ID data to match.";
          }
        }
      }
      case client-duid-id {
        description
          "Client class selection based on the value of the
          received client DUID.";
        leaf duid {
          type dhc6:duid;
          description
            "Client DUID.";
        }
      }
    }
```

```
    }

    /*
     * Augmentations
     */

    augment "/dhc6-srv:dhcpv6-server/dhc6-srv:class-selector" {
      description
        "Augment class selector functions to the DHCPv6 server
        module.";
      container client-classes {
        description
          "Client classes to augment.";
        list class {
          key client-class-name;
          description
            "List of the client class identifiers applicable to
            clients served by this address pool";
          uses client-class-id;
        }
      }
    }

    augment "/dhc6-srv:dhcpv6-server/" +
      "dhc6-srv:network-ranges/dhc6-srv:network-range" {
      description
        "Augment class selector functions to the DHCPv6 server
        network-ranges.";
      leaf-list client-class {
        type leafref {
          path "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
            "class-selector/client-classes/class/client-class-name";
        }
        description
          "Leafrefs to client classes.";
      }
    }

    augment "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
      "network-ranges/dhc6-srv:network-range/dhc6-srv:" +
      "address-pools/dhc6-srv:address-pool" {
      description
        "Augment class selector functions to the DHCPv6 server
        address-pools.";
      leaf-list client-class {
        type leafref {
          path "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
            "class-selector/client-classes/class/client-class-name";
```

```
        }
        description
          "Leafrefs to client classes.";
      }
    }

    augment "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
      "network-ranges/dhc6-srv:network-range/dhc6-srv:" +
      "prefix-pools/dhc6-srv:prefix-pool" {
      description
        "Augment class selector functions to the DHCPv6
        server prefix-pools.";
      leaf-list client-class {
        type leafref {
          path "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
            "class-selector/client-classes/class/client-class-name";
        }
        description
          "Leafrefs to client classes.";
      }
    }
  }
```

Author's Address

Ian Farrer (editor)
Deutsche Telekom AG
TAI, Landgrabenweg 151
53227 Bonn
Germany

Email: ian.farrer@telekom.de