

DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: 2 August 2021

Y. Cui
L. Sun
Tsinghua University
I. Farrer
S.Z. Zechlin
Deutsche Telekom AG
Z. He
Tsinghua University
M. Nowikowski
Internet Systems Consortium
29 January 2021

YANG Data Model for DHCPv6 Configuration
draft-ietf-dhc-dhcpv6-yang-17

Abstract

This document describes YANG data modules for the configuration and management of DHCPv6 servers, relays, and clients.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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 2 August 2021.

Copyright Notice

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

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
1.1.	Scope	3
1.2.	Extensibility of the DHCPv6 Server YANG Module	3
1.2.1.	DHCPv6 Option Definitions	4
1.3.	Terminology	4
2.	DHCPv6 Tree Diagrams	5
2.1.	DHCPv6 Server Tree Diagram	5
2.2.	DHCPv6 Relay Tree Diagram	12
2.3.	DHCPv6 Client Tree Diagram	14
3.	DHCPv6 YANG Modules	18
3.1.	DHCPv6 Server YANG Module	18
3.2.	DHCPv6 Relay YANG Module	35
3.3.	DHCPv6 Client YANG Module	44
3.4.	DHCPv6 Common YANG Module	56
4.	Security Considerations	63
5.	IANA Considerations	64
6.	Acknowledgments	64
7.	Contributors	65
8.	References	65
8.1.	Normative References	65
8.2.	Informative References	67
Appendix A.	Example of Augmenting Additional DHCPv6 Option Definitions	67
Appendix B.	Example Vendor Specific Server Configuration Module	71
Appendix C.	Example definition of class selector configuration	77
	Authors' Addresses	84

1. Introduction

DHCPv6 [[RFC8415](#)] is widely used for supplying configuration and other relevant parameters to clients in IPv6 networks. This document defines YANG modules for the configuration and management of DHCPv6 servers, relays and clients. Separate 'element' modules are defined for each of these. Additionally, a 'common' module contains typedefs and groupings used by all of the element modules.

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 play a part in the normal DHCPv6 message flow.

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 it is easy to augment in 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 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 B](#) 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, research has been carried out into a number of vendor's class selection implementations 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 module. [Appendix C](#) 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. E.g. `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 in the option, so it is not modelled.

Further options definitions can be added by additional YANG modules via augmentation into the relevant element modules from this document. [Appendix A](#) 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 modelling 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
      | +--rw type-code?          uint16
      | +--rw (duid-type)?
      | | +--:(duid-llt)
      | | | +--rw duid-llt-hardware-type?  uint16
      | | | +--rw duid-llt-time?          yang:timeticks
      | | | +--rw duid-llt-link-layer-address?
      | | |         yang:mac-address
      | | +--:(duid-en)
      | | | +--rw duid-en-enterprise-number?  uint32
      | | | +--rw duid-en-identifier?        string
      | | +--:(duid-ll)
      | | | +--rw duid-ll-hardware-type?      uint16
      | | | +--rw duid-ll-link-layer-address?
      | | |         yang:mac-address
      | | +--:(duid-uuid)
      | | | +--rw uuid?                    yang:uuid
      | | +--:(duid-unstructured)
      | |   +--rw data?                    binary
      | +--ro active-duid?                binary
    +--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-option
| | +--rw vendor-specific-information-option-instances*
| | | [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?
| | | dhcpv6-common:timer-seconds32
| +--rw sol-max-rt-option
| | +--rw sol-max-rt-value?
| | | dhcpv6-common:timer-seconds32
| +--rw inf-max-rt-option
| | +--rw inf-max-rt-value?
| | | dhcpv6-common:timer-seconds32
+--rw class-selector
+--rw network-ranges
  +--rw option-set-id* leafref
  +--rw valid-lifetime?
  | dhcpv6-common:timer-seconds32
  +--rw renew-time?
  | dhcpv6-common:timer-seconds32
  +--rw rebind-time?
  | dhcpv6-common:timer-seconds32
  +--rw preferred-lifetime?
  | dhcpv6-common: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?
  | | dhcpv6-common:timer-seconds32
  | +--rw renew-time?
  | | dhcpv6-common:timer-seconds32
  | +--rw rebind-time?
  | | dhcpv6-common:timer-seconds32
  | +--rw preferred-lifetime?
  | | dhcpv6-common: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-count
| |       |
| |       | dhcpv6-common:threshold
| |   +--rw option-set-id*    leafref
| |   +--rw valid-lifetime?
| |       |
| |       | dhcpv6-common:timer-seconds32
| |   +--rw renew-time?
| |       |
| |       | dhcpv6-common:timer-seconds32
| |   +--rw rebind-time?
| |       |
| |       | dhcpv6-common:timer-seconds32
| |   +--rw preferred-lifetime?
| |       |
| |       | dhcpv6-common:timer-seconds32
| |   +--rw rapid-commit?    boolean
| |   +--rw host-reservations
| |       | +--rw host-reservation* [reserved-addr]
| |       |   +--rw client-duid?      binary
| |       |   +--rw reserved-addr
| |       |       |
| |       |       | inet:ipv6-address
| |       |   +--rw option-set-id*    leafref
| |       |   +--rw valid-lifetime?
| |       |       |
| |       |       | dhcpv6-common:timer-seconds32
| |       |   +--rw renew-time?
| |       |       |
| |       |       | dhcpv6-common:timer-seconds32
| |       |   +--rw rebind-time?
| |       |       |
| |       |       | dhcpv6-common:timer-seconds32
| |       |   +--rw preferred-lifetime?
| |       |       |
| |       |       | dhcpv6-common: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?        binary
| |       +--ro ia-id                uint32
| |       +--ro allocation-time?
| |           |
| |           | yang:date-and-time
| |       +--ro last-renew-rebind?
| |           |
| |           | yang:date-and-time
| |       +--ro preferred-lifetime?
| |           |
| |           | dhcpv6-common:timer-seconds32

```



```

| |         +--ro valid-lifetime?
| |         |         dhcpv6-common:timer-seconds32
| |         +--ro lease-t1?
| |         |         dhcpv6-common:timer-seconds32
| |         +--ro lease-t2?
| |         |         dhcpv6-common: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
|       |         dhcpv6-common:threshold
|     +--rw option-set-id*                         leafref
|     +--rw valid-lifetime?
|       |         dhcpv6-common:timer-seconds32
|     +--rw renew-time?
|       |         dhcpv6-common:timer-seconds32
|     +--rw rebind-time?
|       |         dhcpv6-common:timer-seconds32
|     +--rw preferred-lifetime?
|       |         dhcpv6-common:timer-seconds32
|     +--rw rapid-commit?                         boolean
|     +--rw host-reservations
|       | +--rw prefix-reservation* [reserved-prefix]
|       | | +--rw client-duid?                   binary
|       | | +--rw reserved-prefix
|       | | |         inet:ipv6-prefix
|       | | +--rw reserved-prefix-len?          uint8
|       | +--rw option-set-id*                   leafref
|       | +--rw valid-lifetime?
|       | |         dhcpv6-common:timer-seconds32
|       | +--rw renew-time?
|       | |         dhcpv6-common:timer-seconds32
|       | +--rw rebind-time?
|       | |         dhcpv6-common:timer-seconds32
|       | +--rw preferred-lifetime?
|       | |         dhcpv6-common: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?                     binary
|         +--ro ia-id                             uint32

```



```
|
|         +--ro allocation-time?
|         |         yang:date-and-time
|         +--ro last-renew-rebind?
|         |         yang:date-and-time
|         +--ro preferred-lifetime?
|         |         dhcpv6-common:timer-seconds32
|         +--ro valid-lifetime?
|         |         dhcpv6-common:timer-seconds32
|         +--ro lease-t1?
|         |         dhcpv6-common:timer-seconds32
|         +--ro lease-t2?
|         |         dhcpv6-common: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  inet:ipv6-address
| +--ro output
|   +--ro return-message?  string
+---x delete-prefix-lease
| +---w input
| | +---w lease-prefix-to-delete    inet:ipv6-prefix
+--ro output
|   +--ro return-message?  string
```

notifications:

```
+---n address-pool-utilization-threshold-exceeded
| +--ro pool-id?          leafref
| +--ro total-address-count  uint64
| +--ro max-address-count   uint64
| +--ro allocated-address-count  uint64
+---n prefix-pool-utilization-threshold-exceeded
|   {prefix-delegation}?
| +--ro pool-id          leafref
| +--ro max-pd-space-utilization  leafref
| +--ro pd-space-utilization?  uint64
+---n invalid-client-detected
```



```

| +--ro duid?          binary
| +--ro description?   string
+---n decline-received
| +--ro duid?          binary
| +--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?            binary

```

Figure 1: DHCPv6 Server Data Module Structure

Descriptions of important nodes:

- * `enabled`: Enables/disables the function of the 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-bytes) content field. Currently there are four defined types of DUIDs in [\[RFC8415\]](#) and [\[RFC6355\]](#): DUID-LLT, DUID-EN, DUID-LL, and DUID-UUID. DUID-Unknown is used for arbitrary DUID formats which do not follow any of these defined types. 'active-duid' is a read-only field that the server's current DUID can be retrieved from. The DUID definitions are imported from the 'ietf-dhcpv6-common.yang' module.
- * `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 B](#) 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 C](#) 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-ranges', '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 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 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-addresses*
      | |   inet:ipv6-address
      | +--rw link-address?                       binary
      | +--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?                       binary

```



```

    |      +--ro server-duid?          binary
    +--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
  | +---w input
  | | +---w lease-prefix    inet:ipv6-prefix
  | +--ro output
  |   +--ro return-message? string
  +---x clear-client-prefixes
  | +---w input
  | | +---w client-duid     binary
  | +--ro output
  |   +--ro return-message? string
  +---x clear-interface-prefixes
  | +---w input
  | | +---w interface      if:interface-ref
  | +--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 [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 type-code?        uint16
      +--rw (duid-type)?
        | +--:(duid-llt)
        | | +--rw duid-llt-hardware-type?    uint16
        | | +--rw duid-llt-time?             yang:timeticks
        | | +--rw duid-llt-link-layer-address?
        | |      yang:mac-address
        | +--:(duid-en)
        | | +--rw duid-en-enterprise-number?  uint32
        | | +--rw duid-en-identifier?         string
        | +--:(duid-ll)
        | | +--rw duid-ll-hardware-type?      uint16
        | | +--rw duid-ll-link-layer-address?
        | |      yang:mac-address
        | +--:(duid-uuid)
        | | +--rw uuid?                       yang:uuid
        | +--:(duid-unstructured)
        | +--rw data?                         binary
      +--ro active-duid?      binary
      +--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* [user-class-datum-id]
        | |   +--rw user-class-datum-id    uint8
        | |   +--rw user-class-datum?      string
        | +--rw vendor-class-option
        | | +--rw vendor-class-option-instances*
        | |   [enterprise-number]
        | |   +--rw enterprise-number      uint32
        | |   +--rw vendor-class* [vendor-class-datum-id]
        | |     +--rw vendor-class-datum-id    uint8
        | |     +--rw vendor-class-datum?      string
        | +--rw vendor-specific-information-option
        | | +--rw vendor-specific-information-option-instances*
        | |   [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* [iaid]
| +--rw iaid                        uint32
| +--rw ia-na-options
| +--ro lease-state
|   +--ro ia-na-address?           inet:ipv6-address
|   +--ro preferred-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro valid-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro lease-t1?
|   |   dhcpv6-common:timer-seconds32
|   +--ro lease-t2?
|   |   dhcpv6-common:timer-seconds32
|   +--ro allocation-time?         yang:date-and-time
|   +--ro last-renew-rebind?       yang:date-and-time
|   +--ro server-duid?             binary
+--rw ia-ta* [iaid]
| +--rw iaid                        uint32
| +--rw ia-ta-options
| +--ro lease-state
|   +--ro ia-ta-address?           inet:ipv6-address
|   +--ro preferred-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro valid-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro allocation-time?         yang:date-and-time
|   +--ro last-renew-rebind?       yang:date-and-time
|   +--ro server-duid?             binary
+--rw ia-pd* [iaid]
| +--rw iaid                        uint32
| +--rw ia-pd-options
| +--ro lease-state
|   +--ro ia-pd-prefix?            inet:ipv6-prefix
|   +--ro preferred-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro valid-lifetime?
|   |   dhcpv6-common:timer-seconds32
|   +--ro lease-t1?
|   |   dhcpv6-common:timer-seconds32
|   +--ro lease-t2?
|   |   dhcpv6-common:timer-seconds32
|   +--ro allocation-time?         yang:date-and-time
|   +--ro last-renew-rebind?       yang:date-and-time
|   +--ro server-duid?             binary

```



```

    +--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-detected
  |   +--ro iaid          uint32
  |   +--ro description?  string
  +---n retransmission-failed
  |   +--ro failure-type  enumeration
  +---n unsuccessful-status-code
  |   +--ro status-code   uint16
  |   +--ro server-duid   binary
  +---n server-duid-changed
  |   +--ro new-server-duid      binary
  |   +--ro previous-server-duid binary
  +--ro lease-ia-na?
  |   -> /dhcpv6-client/client-if/ia-na/iaid
  +--ro lease-ia-ta?
  |   -> /dhcpv6-client/client-if/ia-ta/iaid
  +--ro lease-ia-pd?
  |   -> /dhcpv6-client/client-if/ia-pd/iaid

```

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 DHCP 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 (of no more than 128-bytes) content field. Currently there are four defined types of DUIDs in [\[RFC8415\]](#): DUID-LLT, DUID-EN, DUID-LL, and DUID-UUID. DUID-Unknown is used for arbitrary DUID formats which do not follow any of these defined types. 'active-duid' is a read-only field that the client's current DUID can be retrieved from. The DUID definitions are imported from the 'ietf-dhcpv6-common.yang' module. DUID is configured under the 'client-if' to allow a client to have different DUIDs for each interface if required.
- * `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 lease 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.yang"
```

```
module ietf-dhcpv6-server {  
  yang-version 1.1;  
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server";  
  prefix "dhcpv6-server";  
  
  import ietf-inet-types {  
    prefix inet;
```



```
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 dhcpv6-common;
  reference
    "To be updated on publication";
}

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

organization "DHC WG";
contact
  "cuiyong@tsinghua.edu.cn
  lh.sunlinh@gmail.com
  ian.farrer@telekom.de
  sladjana.zechlin@telekom.de
  hezihao9512@gmail.com
  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 8513; see
  the RFC itself for full legal notices.";

revision 2021-01-29 {
```



```
    description "Version update for draft -17 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";
}

revision 2021-01-06 {
    description "Version update for draft -16 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";
}

revision 2020-12-22 {
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";
}

revision 2020-12-10 {
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";
}

revision 2020-12-01 {
    description "Version update for draft -12 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";
}

revision 2020-05-26 {
    description "Version update for draft -11 publication and
        to align revisions across the different modules.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
}

revision 2019-12-02 {
    description "Major reworking of the module.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-10";
}

revision 2018-09-04 {
    description "";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

revision 2018-03-04 {
    description "Resolved most issues on the DHC official
        github";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

revision 2017-12-22 {
    description "Resolve most issues on Ian's github.";
```



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

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

  /*
   * Features
   */

  feature prefix-delegation {
    description "Denotes that the server implements DHCPv6 prefix
      delegation.";
  }

  /*
   * 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 of this
        network-range.";
    }
    leaf valid-lifetime {
      type dhcpv6-common:timer-seconds32;
      description "Valid lifetime for the Identity Association
        (IA).";
    }
    leaf renew-time {
      type dhcpv6-common:timer-seconds32;
      description "Renew (T1) time.";
    }
    leaf rebind-time {
      type dhcpv6-common:timer-seconds32;
      description "Rebind (T2) time.";
    }
    leaf preferred-lifetime {
      type dhcpv6-common:timer-seconds32;
```



```
        description "Preferred lifetime for the Identity Association
        (IA).";
    }
    leaf rapid-commit {
        type boolean;
        description "A value of 1 specifies that the pool supports
        client-server exchanges involving two messages.";
    }
}

grouping lease-information {
    description "Binding information for each client that has
    been allocated an IPv6 address or prefix.";
    leaf client-duid {
        type binary;
        description "Client DUID.";
    }
    leaf iaid {
        type uint32;
        mandatory true;
        description "Client's IAID";
    }
    leaf allocation-time {
        type yang:date-and-time;
        description "Time and date that the lease was made.";
    }
    leaf last-renew-rebind {
        type yang:date-and-time;
        description "Time of the last successful renew or
        rebind.";
    }
    leaf preferred-lifetime {
        type dhcpv6-common:timer-seconds32;
        description "The preferred lifetime expressed in
        seconds.";
    }
    leaf valid-lifetime {
        type dhcpv6-common:timer-seconds32;
        description "The valid lifetime for the leased prefix
        expressed in seconds.";
    }
    leaf lease-t1 {
        type dhcpv6-common: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 dhcpv6-common: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.";
}
}

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 "RFC8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6)";
  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 "RFC8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6)";
  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 "RFC8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6)";
    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 "RFC8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6)";
    container info-refresh-time-option {
      description "OPTION_INFORMATION_REFRESH_TIME (32)
        Information Refresh Time option container.";
      leaf info-refresh-time {
        type dhcpv6-common: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.";
    reference "RFC8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6)";
    container sol-max-rt-option {
      description "OPTION_SOL_MAX_RT (82) sol max rt option
        container.";
      leaf sol-max-rt-value {
        type dhcpv6-common:timer-seconds32;
        description "sol max rt value";
      }
    }
  }
}
```



```
grouping inf-max-rt-option-group {
  description "OPTION_INF_MAX_RT (83) inf max rt option.";
  reference "RFC8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6)";
  container inf-max-rt-option {
    description "OPTION_INF_MAX_RT (83) inf max rt option
      container.";
    leaf inf-max-rt-value {
      type dhcpv6-common:timer-seconds32;
      description "inf max rt value";
    }
  }
}

/*
 * Data Nodes
 */

container dhcpv6-server {
  description "Configuration nodes for the DHCPv6 server.";
  leaf enabled {
    description "Enables the DHCP server function.";
    type boolean;
    default true;
  }
  container server-duid {
    description "DUID of the server.";
    uses dhcpv6-common:duid;
  }
  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.";
    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 dhcpv6-common:auth-option-group;
    uses server-unicast-option-group;
    uses dhcpv6-common:status-code-option-group;
    uses dhcpv6-common:rapid-commit-option-group;
    uses dhcpv6-common:vendor-specific-information-option-group;
    uses reconfigure-message-option-group;
    uses dhcpv6-common: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 that they
        are 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 "Equivalent to subnet ID.";
        }
        leaf description {
            type string;
```



```
    mandatory true;
    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 "Start IPv6 address for the pool.";
      }
      leaf end-address {
        type inet:ipv6-address-no-zone;
        mandatory true;
        description "End IPv6 address for the pool.";
      }
      leaf max-address-count {
        type dhcpv6-common:threshold;
        mandatory true;
        description "Maximum number of addresses that can
          be simultaneously allocated from this pool.";
      }
    }
    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 binary;
            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;
    mandatory true;
    description "Length of the prefixes that will be
        delegated to clients.";
}
leaf max-pd-space-utilization {
    type dhcpv6-common:threshold;
    mandatory true;
    description "Maximum percentage utilization of the
        prefix pool in this pool.";
}
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 binary;
            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 for 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 configured threshold.";
    leaf pool-id {
        type leafref {
            path "/dhcpv6-server/network-ranges/network-range/" +
                "address-pools/address-pool/pool-id";
        }
        description "Leafref to the address pool which the notification
            is being generated for.";
    }
    leaf total-address-count {
        type uint64;
        mandatory true;
        description "Count of the total addresses in the pool.";
    }
}
```



```
    }
    leaf max-address-count {
        type uint64;
        mandatory true;
        description "Maximum count of addresses that can be allocated
            in the pool. This value may be less than count of total
            addresses.";
    }
    leaf allocated-address-count {
        type uint64;
        mandatory true;
        description "Count of allocated addresses in the pool.";
    }
}

notification prefix-pool-utilization-threshold-exceeded {
    if-feature prefix-delegation;
    description "Notification sent when the prefix pool
        utilization exceeds the configured threshold.";
    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 max-pd-space-utilization {
        type leafref {
            path "/dhcpv6-server/network-ranges/network-range/" +
                "prefix-pools/prefix-pool/max-pd-space-utilization";
        }
        mandatory true;
        description "PD space utilization threshold.";
    }
    leaf pd-space-utilization {
        type uint64;
        description "Current PD space utilization";
    }
}

notification invalid-client-detected {
    description "Notification sent when the server detects an
        invalid client.";
    leaf duid {
        type binary;
        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 binary;
    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 binary;
    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 inet:ipv6-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;
  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 inet:ipv6-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.yang"
```

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

  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 dhcpv6-common;
    reference
      "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
    "cuiyong@tsinghua.edu.cn
    lh.sunlinh@gmail.com
    ian.farrer@telekom.de"
```


sladjana.zechlin@telekom.de
hezihao9512@gmail.com
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 8513](#); see the RFC itself for full legal notices.";

```
revision 2021-01-29 {  
  description "Version update for draft -17 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";  
}
```

```
revision 2021-01-06 {  
  description "Version update for draft -16 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";  
}
```

```
revision 2020-12-22 {  
  description "Version update for draft -13 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";  
}
```

```
revision 2020-12-10 {  
  description "Version update for draft -13 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";  
}
```

```
revision 2020-12-01 {  
  description "Version update for draft -12 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";  
}
```

```
revision 2020-05-26 {  
  description "Version update for draft -11 publication and
```



```
        to align revisions across the different modules.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
}

revision 2019-09-20 {
    description "";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-10";
}

revision 2018-03-04 {
    description "Resolved most issues on the DHC official
        github";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

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

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

/*
 * Features
 */

feature prefix-delegation {
    description "Enable if the relay functions as a delegating router
        for DHCPv6 prefix delegation.";
}

/*
 * 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 client.";
  }
  leaf client-duid {
    type binary;
    description "DUID of the leasing client.";
  }
  leaf server-duid {
    type binary;
    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 "RFC8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6)";
  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.";
  leaf enabled {
    description "Globally enables the DHCP relay function.";
    type boolean;
    default true;
  }
  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 {
      description "Enables the DHCP relay function for this
        interface.";
      type boolean;
      default true;
    }
  }
  leaf-list destination-addresses {
    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 binary {
            length "0..16";
        }
        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 dhcpv6-common:auth-option-group;
        uses dhcpv6-common: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 and 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;
    description "Clears an entry for an active delegated prefix
        from the relay.";
    input {
        leaf lease-prefix {
            type inet:ipv6-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;
    description "Clears all active prefix entries for a single
        client.";
    input {
        leaf client-duid {
            type binary;
            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;
```



```
description "Clears all delegated prefix bindings from an
  interface on the relay.";
input {
  leaf interface {
    type if:interface-ref;
    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.yang"
```

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

  import ietf-inet-types {
    prefix inet;
    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 dhcpv6-common;
    reference
      "To be updated on publication";
  }
}
```



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

organization "DHC WG";
contact
  "cuiyong@tsinghua.edu.cn
  wanh13@mails.tsinghua.edu.cn
  lh.sunlinh@gmail.com
  ian.farrer@telekom.de
  sladjana.zechlin@telekom.de
  hezihao9512@gmail.com
  godfryd@isc.org";

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

  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 8513; see
  the RFC itself for full legal notices."

revision 2021-01-29 {
  description "Version update for draft -17 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";
}

revision 2021-01-06 {
  description "Version update for draft -16 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";
}

revision 2020-12-22 {
  description "Version update for draft -13 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";
}
```



```
revision 2020-12-10 {
  description "Version update for draft -13 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";
}

revision 2020-12-01 {
  description "Version update for draft -12 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";
}

revision 2020-05-26 {
  description "Version update for draft -11 publication and
    to align revisions across the different modules.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
}

revision 2019-09-20 {
  description "";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-10";
}

revision 2018-09-04 {
  description "";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

revision 2018-03-04 {
  description "Resolved most issues on the DHC official github";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

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

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

/*
 * 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 "RFC8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6)";
    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 "RFC8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6)";
    container user-class-option {
        description "OPTION_USER_CLASS (15) User Class Option
            container.";
        list user-class-data {
            key user-class-datum-id;
            min-elements 1;
            description "The user classes of which the client
                is a member.";
            leaf user-class-datum-id {
                type uint8;
                description "User class datum ID";
            }
            leaf user-class-datum {
                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 "RFC8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6)";
  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 {
        key vendor-class-datum-id;
        description "The vendor classes of which the client is
          a member.";
        leaf vendor-class-datum-id {
          type uint8;
          description "Vendor class datum ID";
        }
        leaf vendor-class-datum {
          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 {
    description "Globally enables the DHCP client function.";
  }
}
```



```
    type boolean;
    default true;
}
list client-if {
  key if-name;
  description "The list of interfaces that the client will be
    requesting DHCPv6 configuration for.";
  leaf if-name {
    type if:interface-ref;
    mandatory true;
    description "Reference to the interface entry that
      the requested configuration is relevant to.";
  }
  leaf enabled {
    description "Enables the DHCP client function for this
      interface.";
    type boolean;
    default true;
  }
}
uses dhcpv6-common:duid;
container client-configured-options {
  description "Definitions for DHCPv6 options that can be be
    sent by the client are augmented to this location from
    other YANG modules as required.";
  uses option-request-option-group;
  uses dhcpv6-common:status-code-option-group;
  uses dhcpv6-common:rapid-commit-option-group;
  uses user-class-option-group;
  uses vendor-class-option-group;
  uses dhcpv6-common:vendor-specific-information-option-group;
  uses dhcpv6-common:reconfigure-accept-option-group;
}
list ia-na {
  key iaid;
  description "Configuration relevant for an IA_NA.";
  reference "RFC8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6).";
  leaf iaid {
    type uint32;
    description "A unique identifier for this IA_NA.";
  }
  container ia-na-options {
    description "An augmentation point for additional options
      that the client will 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 dhcpv6-common:timer-seconds32;
        description "The preferred lifetime for the leased
            address expressed in units of seconds.";
    }
    leaf valid-lifetime {
        type dhcpv6-common:timer-seconds32;
        description "The valid lifetime for the leased address
            expressed in units of seconds.";
    }
    leaf lease-t1 {
        type dhcpv6-common: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 dhcpv6-common: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 binary;
        description "DUID of the leasing server.";
    }
}

list ia-ta {
    key iaid;
    description "Configuration relevant for an IA_TA.";
    reference "RFC8415: Dynamic Host Configuration Protocol for
```



```
    IPv6 (DHCPv6).";
    leaf iaid {
        type uint32;
        description "The unique identifier for this IA_TA.";
    }
    container ia-ta-options {
        description "An augmentation point for additional options
            that the client will 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 dhcpv6-common:timer-seconds32;
            description "The preferred lifetime for the leased
                address expressed in units of seconds.";
        }
        leaf valid-lifetime {
            type dhcpv6-common:timer-seconds32;
            description "The valid lifetime for the leased address
                expressed in units of 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 binary;
            description "DUID of the leasing server.";
        }
    }
}
list ia-pd {
    key iaid;
    description "Configuration relevant for an IA_PD.";
    reference "RFC8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6).";
```



```
leaf iaid {
  type uint32;
  description "The unique identifier for this IA_PD.";
}
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 dhcpv6-common:timer-seconds32;
    description "The preferred lifetime for the leased prefix
      expressed in units of seconds.";
  }
  leaf valid-lifetime {
    type dhcpv6-common:timer-seconds32;
    description "The valid lifetime for the leased prefix
      expressed in units of seconds.";
  }
  leaf lease-t1 {
    type dhcpv6-common: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 dhcpv6-common: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 binary;
        description "DUID of the delegating server.";
    }
}
}
uses message-statistics;
}
}

/*
 * Notifications
 */

notification invalid-ia-detected {
    description "Notification sent when the identity association
        of the client can be proved to be invalid. Possible conditions
        include a duplicate or otherwise illegal address.";
    leaf iaaid {
        type uint32;
        mandatory true;
        description "IAID";
    }
    leaf description {
        type string;
        description "Description of the event.";
    }
}

notification retransmission-failed {
    description "Notification sent when the retransmission mechanism
        defined in \[RFC8415\] is unsuccessful.";
    leaf failure-type {
        type enumeration {
            enum "MRC-exceeded" {
                description "Maximum retransmission count exceeded.";
            }
            enum "MRD-exceeded" {
                description "Maximum retransmission duration exceeded.";
            }
        }
        mandatory true;
        description "Description of the failure.";
    }
}
```



```
notification unsuccessful-status-code {
  description "Notification sent when the client receives a message
    that includes an unsuccessful Status Code option.";
  leaf status-code {
    type uint16;
    mandatory true;
    description "Unsuccessful status code received by a client.";
  }
  leaf server-duid {
    type binary;
    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.";
  leaf new-server-duid {
    type binary;
    mandatory true;
    description "DUID of the new server.";
  }
  leaf previous-server-duid {
    type binary;
    mandatory true;
    description "DUID of the previous server.";
  }
  leaf lease-ia-na {
    type leafref {
      path "/dhcpv6-client/client-if/ia-na/iaid";
    }
    description "Reference to the IA_NA lease.";
  }
  leaf lease-ia-ta {
    type leafref {
      path "/dhcpv6-client/client-if/ia-ta/iaid";
    }
    description "Reference to the IA_TA lease.";
  }
  leaf lease-ia-pd {
    type leafref {
      path "/dhcpv6-client/client-if/ia-pd/iaid";
    }
    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.yang"  
  
module ietf-dhcpv6-common {  
  yang-version 1.1;  
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcpv6-common";  
  prefix "dhcpv6-common";  
  
  import ietf-yang-types {  
    prefix yang;  
    reference  
      "RFC 6991: Common YANG Data Types";  
  }  
  
  organization "DHC WG";  
  contact  
    "yong@csnet1.cs.tsinghua.edu.cn  
    lh.sunlinh@gmail.com  
    ian.farrer@telekom.de  
    sladjana.zechlin@telekom.de  
    hezihao9512@gmail.com";  
  
  description "This YANG module defines common components  
    used for the configuration and management of DHCPv6.  
  
    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 8513; see  
    the RFC itself for full legal notices."  
  
  revision 2021-01-29 {  
    description "Version update for draft -17 publication.";  
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";
```



```
}

revision 2021-01-06 {
  description "Version update for draft -16 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";
}

revision 2020-12-22 {
  description "Version update for draft -13 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";
}

revision 2020-12-10 {
  description "Version update for draft -13 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";
}

revision 2020-12-01 {
  description "Version update for draft -12 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";
}

revision 2020-05-26 {
  description "Version update for draft -11 publication and
    to align revisions across the different modules.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
}

revision 2018-09-04 {
  description "";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang";
}

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

typedef threshold {
  type union {
    type uint16 {
      range 0..100;
    }
    type enumeration {
      enum "disabled" {
        description "No threshold";
      }
    }
  }
}
```



```
    }
    description "Threshold value in percent";
  }

typedef timer-seconds32 {
  type uint32 {
    range "1..4294967295";
  }
  units "seconds";
  description
    "Timer value type, in seconds (32-bit range).";
}

/*
 * Groupings
 */

grouping duid {
  description "Each server and client has only one DUID (DHCP
    Unique Identifier). The DUID here identifies a unique
    DHCPv6 server for clients. DUID consists of a two-octet
    type field and an arbitrary length (no more than 128 bytes)
    content field. Currently there are four defined types of
    DUIDs in RFC8415 and RFC6355 - DUID-LLT, DUID-EN, DUID-LL
    and DUID-UUID. DUID-unstructured represents DUIDs which
    do not follow any of the defined formats.";
  reference "RFC8415: Section 11 and RFC6355: Section 4";
  leaf type-code {
    type uint16;
    default 65535;
    description "Type code of this DUID.";
  }
  choice duid-type {
    default duid-unstructured;
    description "Selects the format of the DUID.";
    case duid-llt {
      description "DUID Based on Link-layer Address Plus Time
        (Type 1 - DUID-LLT).";
      reference "RFC8415 Section 11.2";
      leaf duid-llt-hardware-type {
        type uint16;
        description "Hardware type as assigned by IANA (RFC826).";
      }
    }
    leaf duid-llt-time {
      type yang:timeticks;
      description "The time that the DUID is generated
        represented in seconds since midnight (UTC),
        January 1, 2000, modulo 2^32.";
```



```
    }
    leaf duid-llt-link-layer-address {
        type yang:mac-address;
        description "Link-layer address as described in RFC2464";
    }
}
case duid-en {
    description "DUID Assigned by Vendor Based on Enterprise
        Number (Type 2 - DUID-EN).";
    reference "RFC8415 Section 11.3";
    leaf duid-en-enterprise-number {
        type uint32;
        description "Vendor's registered Private Enterprise Number
            as maintained by IANA.";
    }
    leaf duid-en-identifier {
        type string;
        description "Identifier, unique to the device.";
    }
}
case duid-ll {
    description "DUID Based on Link-layer Address
        (Type 3 - DUID-LL).";
    reference "RFC8415 Section 11.4";
    leaf duid-ll-hardware-type {
        type uint16;
        description "Hardware type, as assigned by IANA (RFC826).";
    }
    leaf duid-ll-link-layer-address {
        type yang:mac-address;
        description "Link-layer address, as described in RFC2464";
    }
}
case duid-uuid {
    description "DUID Based on Universally Unique Identifier
        (Type 4 - DUID-UUID).";
    reference "RFC6335 Definition of the UUID-Based Unique
        Identifier";
    leaf uuid {
        type yang:uuid;
        description "A Universally Unique Identifier in the string
            representation, defined in RFC4122. The canonical
            representation uses lowercase characters.";
    }
}
case duid-unstructured {
    description "DUID which does not follow any of the other
        structures, expressed as bytes.";
```



```
        leaf data {
            type binary;
            description "The bits to be used as the identifier.";
        }
    }
}
leaf active-duid {
    type binary;
    config "false";
    description "The DUID which is currently in use.";
}
}

grouping auth-option-group {
    description "OPTION_AUTH (11) Authentication Option.";
    reference "RFC8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6)";
    container auth-option {
        description "OPTION_AUTH (11) Authentication Option
            container.";
        leaf protocol {
            type uint8;
            description "The authentication protocol used in this
                Authentication option.";
        }
        leaf algorithm {
            type uint8;
            description "The algorithm used in the authentication
                protocol.";
        }
        leaf rdm {
            type uint8;
            description "The replay detection method 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 "RFC8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6)";
  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. MUST NOT be null-terminated.";
    }
  }
}

grouping rapid-commit-option-group {
  description "OPTION_RAPID_COMMIT (14) Rapid Commit Option.";
  reference "RFC8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6)";
  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 "RFC8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6)";
  container vendor-specific-information-option {
    description "OPTION_VENDOR_OPTS (17) Vendor-specific
      Information Option container.";
    list vendor-specific-information-option-instances {
      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.";
    }
    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;
            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 "RFC8415: Dynamic Host Configuration Protocol
                for IPv6 (DHCPv6)";
    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. E.g., 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:
(dhcpv6-server/network-ranges/network-range/ address-pools/
address-pool/active-leases)
- * Information the relay holds about clients with active leases:
(dhcpv6-relay/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 registers the following YANG modules in the "YANG Module Names" registry [[RFC6020](#)].

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

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

6. Acknowledgments

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

7. Contributors

The following individuals contributed to this effort:

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

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

Bernie Volz
Cisco Systems, Inc.
1414 Massachusetts Ave
Boxborough, MA 01719
USA
Email: volz@cisco.com

8. References

8.1. Normative References

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

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](#). 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 which specifies the DHCP option format.
- * 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).

<CODE BEGINS> file "ietf-example-dhcpv6-options-sip-server.yang"

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

  import ietf-inet-types {
    prefix inet;
  }

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

  organization "DHC WG";
  contact
    "ian.farrer@telekom.de
    godfryd@isc.org";

  description "This YANG module contains DHCPv6 options defined
    in RFC3319 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 8513; see
    the RFC itself for full legal notices."

  revision 2021-01-29 {
    description "Version update for draft -17 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";
  }

  revision 2021-01-06 {
    description "Version update for draft -16 publication.";
```



```
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";
  }

  revision 2020-12-22 {
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";
  }

  revision 2020-12-10 {
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";
  }

  revision 2020-12-01 {
    description "Version update for draft -12 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";
  }

  revision 2020-05-26 {
    description "Version update for draft -11 publication and
      to align revisions across the different modules.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
  }

  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 "RFC3319: 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 "RFC3319: 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 "/dhcpv6-server:dhcpv6-server/dhcpv6-server:option-sets/" +

    "dhcpv6-server: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;
    }
}
<CODE ENDS>
```


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. E.g. for options which will be augmented into the ietf-dhcpv6-server module, in many cases, these will be augmented to:

```
'/dhcpv6-server:dhcpv6-server/dhcpv6-server:option-sets/\ dhcpv6-server: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:

```
'/dhcpv6-server:dhcpv6-server/dhcpv6-server:network-ranges/\ dhcpv6-server:network-range/dhcpv6-server:prefix-pools/\ dhcpv6-server:prefix-pool'
```

Appendix B. 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 PostgreSQL. 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.

<CODE BEGINS> file "ietf-example-dhcpv6-server-config.yang"

```
module ietf-example-dhcpv6-server-config {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:" +
    "ietf-example-dhcpv6-server-config";
  prefix "dhcpv6-server-config";

  import ietf-inet-types {
    prefix inet;
```



```
}

import ietf-interfaces {
  prefix if;
}

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

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

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
  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 8513; see
the RFC itself for full legal notices.";

revision 2021-01-29 {
  description "Version update for draft -17 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";
}

revision 2021-01-06 {
  description "Version update for draft -16 publication.";
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";
}

revision 2020-12-22 {
```



```
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";
}

revision 2020-12-10 {
    description "Version update for draft -13 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";
}

revision 2020-12-01 {
    description "Version update for draft -12 publication.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";
}

revision 2020-05-26 {
    description "Version update for draft -11 publication and
        to align revisions across the different modules.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
}

revision 2019-06-04 {
    description "";
    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 that the server will
        listen for incoming messages on. 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) that the server
        will listen for incoming messages on.";
    }
  }
}
leaf-list interfaces-config {
  type if:interface-ref;
  default "if:interfaces/if:interface/if:name";
  description "A leaf list to denote which one or more
    interfaces the server should listen on.";
}
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 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 "/dhcpv6-server:dhcpv6-server/dhcpv6-server:vendor-config"
{
    description "Augment the server specific YANG to the
        ietf-dhcpv6-server module.";
    uses dhcpv6-server-config:config;
}
}
<CODE ENDS>

```

Appendix C. 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), packets 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 can 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.

<CODE BEGINS> file "ietf-example-dhcpv6-class-selector.yang"

```
module ietf-example-dhcpv6-class-selector {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:" +
    "ietf-example-dhcpv6-class-selector";
  prefix "dhcpv6-class-selector";

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-dhcpv6-common {
    prefix dhcpv6-common;
  }

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

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


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

description "This 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 8513](#); see the RFC itself for full legal notices.";

```
revision 2021-01-29 {  
  description "Version update for draft -17 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-17";  
}
```

```
revision 2021-01-06 {  
  description "Version update for draft -16 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-16";  
}
```

```
revision 2020-12-22 {  
  description "Version update for draft -13 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-15";  
}
```

```
revision 2020-12-10 {  
  description "Version update for draft -13 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-13";  
}
```

```
revision 2020-12-01 {  
  description "Version update for draft -12 publication.";  
  reference "I-D: draft-ietf-dhc-dhcpv6-yang-12";  
}
```

```
revision 2020-05-26 {
```



```
    description "Version update for draft -11 publication and
      to align revisions across the different modules.";
    reference "I-D: draft-ietf-dhc-dhcpv6-yang-11";
  }

  revision 2019-06-13 {
    description "";
    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.";
    uses dhcpv6-common:duid;
}
}
}

/*
 * Augmentations
 */

augment "/dhcpv6-server:dhcpv6-server/dhcpv6-server: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 dhcpv6-class-selector:client-class-id;
        }
    }
}

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



```
        "class-selector/client-classes/class/client-class-name";
    }
    description "Leafrefs to client classes.";
}
}

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

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

Authors' Addresses

Yong Cui
Tsinghua University
Beijing
100084
P.R. China

Phone: +86-10-6260-3059
Email: cuiyong@tsinghua.edu.cn

Linhui Sun
Tsinghua University
Beijing
100084
P.R. China

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

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

Email: ian.farrer@telekom.de

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

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

Michal Nowikowski
Internet Systems Consortium
Gdansk
Poland

Email: godfryd@isc.org

