

DHC Work Group  
Internet-Draft  
Intended status: Standards Track  
Expires: June 19, 2020

I. Farrer  
Deutsche Telekom AG  
Naveen. Kottapalli  
Benu Networks  
M. Hunek  
Technical University of Liberec  
Richard. Patterson  
December 17, 2019

DHCPv6 Prefix Delegating relay  
draft-fkhp-dhc-dhcpv6-pd-relay-requirements-03

Abstract

Operational experience with DHCPv6 prefix delegation has shown that when the DHCPv6 relay function is not co-located with the DHCPv6 server function, issues such as timer synchronization between the DHCP functional elements, rejection of client's messages by the relay, and other problems have been observed. These problems can result in prefix delegation failing or traffic to/from clients addressed from the delegated prefix being unroutable. Although [RFC8415] mentions this deployment scenario, it does not provide necessary detail on how the relay element should behave when used with PD.

This document describes functional requirements for a DHCPv6 PD relay when used for relaying prefixes delegated by a separate DHCPv6 server.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on June 19, 2020.

## Copyright Notice

Copyright (c) 2019 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 . . . . .	2
2. Terminology . . . . .	3
2.1. General . . . . .	3
2.2. Topology . . . . .	4
2.3. Requirements Language . . . . .	5
3. Problems Observed with Existing Delegating Relays Implementations . . . . .	5
3.1. DHCP Messages not being Forwarded by the Delegating relay . . . . .	5
3.2. Delegating Relay Loss of State on Reboot . . . . .	5
3.3. Multiple Simultaneous Delegated Prefixes for a Single DUID on a Single Client . . . . .	6
3.4. Dropping Messages from Devices with Duplicate MAC addresses and DUIDs . . . . .	6
4. Requirements for Delegating Relays . . . . .	6
4.1. General Requirements . . . . .	6
4.2. Routing Requirements . . . . .	7
4.3. Service Continuity Requirements . . . . .	8
4.4. Operational Requirements . . . . .	8
5. Acknowledgements . . . . .	9
6. IANA Considerations . . . . .	9
7. Security Considerations . . . . .	9
8. References . . . . .	9
8.1. Normative References . . . . .	9
8.2. Informative References . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

For internet service providers that offer native IPv6 access with prefix delegation to their customers, a common deployment architecture is to have a DHCPv6 relay agent function located in the

ISP's Layer-3 customer edge device and separate, centralized DHCPv6 server infrastructure. [RFC8415] describes the functionality of a DHCPv6 relay and Section 19.1.3 mentions the deployment scenario, but does not provide detail for all of the functional requirements that the relay needs to fulfill to operate deterministically in this deployment scenario.

A DHCPv6 relay agent for prefix delegation is a function commonly implemented in routing devices, but implementations vary in their functionality and client/server inter-working. This can result in operational problems such as messages not being forwarded by the relay or unreachability of the delegated prefixes. This document provides a set of requirements for devices implementing a relay function for use with prefix delegation.

The mechanisms for the redistribution of remote routes learnt via DHCP PD is out of scope of the document. Multi-hop relaying is also not considered as the functionality is solely required by a DHCP relay agent that is co-located with the first-hop router that the DHCPv6 client requesting the prefix is connected to.

The behavior defined in [RFC7283] is also applicable for DHCPv6-PD-relay deployments.

## 2. Terminology

### 2.1. General

This document uses the terminology defined in [RFC8415], however when defining the functional elements for prefix delegation [RFC8415], Section 4.2 defines the term 'delegating router' as:

"The router that acts as a DHCP server and responds to requests for delegated prefixes."

This document is concerned with deployment scenarios in which the DHCPv6 relay and DHCPv6 server functions are separated, so the term 'delegating router' is not used. Instead, a new term is introduced to describe the relaying function:

**Delegating relay** A delegating relay acts as an intermediate device, forwarding DHCPv6 messages containing IA\_PD/IAPREFIX options between the client and server. The delegating relay does not implement a DHCPv6 server function. The delegating relay is also responsible for routing traffic for the delegated prefixes.

Where the term 'relay' is used on its own within this document, it should be understood to be a delegating relay, unless specifically stated otherwise.

[RFC8415] defines the 'DHCP server', (or as 'server') as:

"A node that responds to requests from clients. It may or may not be on the same link as the client(s). Depending on its capabilities, if it supports prefix delegation it may also feature the functionality of a delegating router.

This document serves the deployment cases where DHCPv6 server is not located on the same link as the client (necessitating the delegating relay). The server supports prefix delegation and is capable of leasing prefixes to clients, but is not responsible for other functions required of a delegating router, such as managing routes for the delegated prefixes.

The term 'requesting router' has previously been used to describe the DHCP client requesting prefixes for use. This document adopts the [RFC8415] terminology and uses 'DHCP client' or 'client' interchangeably for this element.

## 2.2. Topology

The following diagram shows the deployment topology relevant to this document.

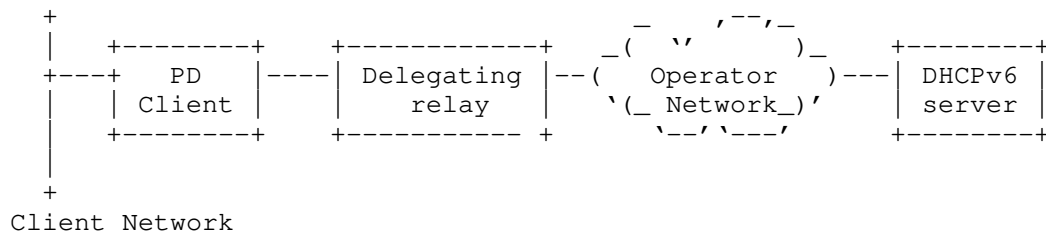


Figure 1

The client request prefixes via the client facing interface of the delegating relay. The resulting prefixes will be used for addressing the client network. The delegating relay is responsible for forwarding DHCP messages, including prefix delegation requests and responses between the client and server. Messages are forwarded from the delegating relay to the server using multicast or unicast via the operator network facing interface.

The delegating relay provides the operator's Layer-3 edge towards the client and is responsible for routing traffic to and from clients connected to the client network using addresses from the delegated prefixes.

### 2.3. 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. This document uses these keywords not strictly for the purpose of interoperability, but rather for the purpose of establishing industry-common baseline functionality. As such, the document points to several other specifications (preferably in RFC or stable form) to provide additional guidance to implementers regarding any protocol implementation required to produce a DHCP relaying router that functions successfully with prefix delegation.

### 3. Problems Observed with Existing Delegating Relays Implementations

The following sections of the document describe problems that have been observed with delegating relay implementations in commercially available devices.

#### 3.1. DHCP Messages not being Forwarded by the Delegating relay

Delegating relay implementations have been observed not to forward messages between the client and server. This generally occurs if a client sends a message which is unexpected by the delegating relay. For example, the delegating router already has an active PD lease entry for an existing client on a port. A new client is connected to this port and sends a solicit message. The delegating relay then drops the solicit messages until it receives either a DHCP release message from the original client, or the existing lease times out. This causes a particular problem when a client device needs to be replaced due to a failure.

In addition to dropping messages, in some cases the delegating relay will generate error messages and send them to the client, e.g. 'NoBinding' messages being sent in the event that the delegating relay does not have an active delegated prefix lease.

#### 3.2. Delegating Relay Loss of State on Reboot

For proper routing of client's traffic, the delegating relay requires a corresponding routing table entry for each active prefix delegated to a connected client. A delegating router which does not store this

state persistently across reboots will not be able to forward traffic to client's delegated leases until the state is re-established through new DHCP messages.

### 3.3. Multiple Simultaneous Delegated Prefixes for a Single DUID on a Single Client

[RFC8415] allows for a client to include more than one instance of `OPTION_IA_PD` in messages in order to request multiple prefix delegations by the server. If configured for this, the server supplies one instance of `OPTION_IAPREFIX` for each received instance of `OPTION_IA_PD`, each containing information for a different delegated prefix.

In some delegating relay implementations, only a single delegated prefix per-DUID is supported. In those cases only one IPv6 route for only one of the delegated router is installed; meaning that other prefixes delegated to a client are unreachable.

### 3.4. Dropping Messages from Devices with Duplicate MAC addresses and DUIDs

It is an unfortunate operational reality that client devices with duplicate MAC addresses and/or DUIDs exist and have been deployed. In this situation, the operational costs of locating and swapping out such devices are prohibitive.

Delegating relays have been observed to restrict forwarding client messages originating from one client DUID to a single interface. In this case if the same client DUID appears from a second client on another interface while there is already an active lease, messages originating from the second client are dropped causing the second client to be unable to obtain a prefix delegation.

## 4. Requirements for Delegating Relays

To resolve the problems described in Section 3 the following section of the document describes a set of functional requirements for the delegating relay.

### 4.1. General Requirements

G-1: The delegating router **MUST** forward messages bidirectionally between the client and server without changing the contents of the message.

- G-2: As described in Section 16 of [RFC8415], in the event that a received message contains a DHCPv6 option which the relay does not implement, the message **MUST** be forwarded.
- G-3: The relay **MUST** allow for multiple prefixes to be delegated for the same client IA\_PD. These delegations may have different lifetimes.
- G-4: The relay **MUST** allow for multiple prefixes with separate IA\_PDs to be delegated to a single client connected to a single interface, identified by its DHCPv6 Client Identifier (DUID).
- G-5: The relay **MUST** allow the same client identifier (DUID) to have active delegated prefix leases on more than one interface simultaneously. This is to allow client devices with duplicate DUIDs to function on separate broadcast domains.
- G-6: The maximum number of simultaneous prefixes delegated to a single client **MUST** be configurable.
- G-7: The relay **MUST** implement a mechanism to limit the maximum number of active prefix delegations on a single port for all client identifiers and IA\_PDs. This value **SHOULD** be configurable.
- G-8: The delegating relay **MUST** synchronize the lifetimes of active prefix delegation leases with server.

#### 4.2. Routing Requirements

- R-1: The relay **MUST** maintain a local routing table that is dynamically updated with prefixes and the associated next-hops as they are delegated to clients. When a delegated prefix is released or expires, the associated route **MUST** be removed from the relay's routing table.
- R-2: The relay **MUST** provide a mechanism to dynamically update access control lists permitting ingress traffic sourced from clients' delegated prefixes. This is to implement anti-spoofing as described in [BCP38].
- R-3: The relay **MAY** provide a mechanism to dynamically advertise delegated prefixes into an routing protocol as they are learnt. When a delegated prefix is released or expires, the delegated route **MUST** be withdrawn from the routing protocol.

The mechanism using which the routes are inserted and deleted is out of the scope of this document.

#### 4.3. Service Continuity Requirements

- S-1: In the event that the relay is restarted, active client prefix delegations will be lost. This may result in clients becoming unreachable. In order to mitigate this problem, it is RECOMMENDED that the relay implements either of the following:

The relay MAY implement DHCPv6 bulk lease query as defined in [RFC5460].

The relay SHOULD store active prefix delegations in persistent storage so they can be re-read after the reboot.

- S-2: If a client's next-hop link-local address becomes unreachable (e.g., due to a link-down event on the relevant physical interface), routes for the client's delegated prefixes MUST be retained by the delegating relay unless they are released or removed due to expiring DHCP timers. This is to re-establish routing for the delegated prefix if the client next-hop becomes reachable without the relay needing to be re-learned.
- S-3: The relay MAY implement DHCPv6 active lease query as defined in [RFC7653] to keep the local lease database in sync with the DHCPv6 server.

#### 4.4. Operational Requirements

- O-1: The relay SHOULD implement an interface allowing the operator to view the active delegated prefixes. This SHOULD provide information about the delegated lease and client details such as client identifier, next-hop address, connected interface, and remaining lifetimes.
- O-2: The relay SHOULD provide a method for the operator to clear active bindings for an individual lease, client or all bindings on a port.
- O-3: To facilitate troubleshooting of operational problems between the delegating relay and other elements, it is RECOMMENDED



that the delegating relay's system time is synchronised with the network.

## 5. Acknowledgements

The authors of this document would like to thank Bernie Volz for his valuable comments.

## 6. IANA Considerations

This memo includes no request to IANA.

## 7. Security Considerations

If the delegating relay implements [BCP38] filtering, then the filtering rules will need to be dynamically updated as delegated prefixes are leased.

[RFC8213] describes a method for securing traffic between the relay agent and server by sending DHCP messages over an IPSec tunnel. In this case the IPSec tunnel is functionally the server-facing interface and DHCPv6 message snooping can be carried out as described. It is RECOMMENDED that this is implemented by the delegating relay.

## 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>>.
- [RFC5460] Stapp, M., "DHCPv6 Bulk Leasequery", RFC 5460, DOI 10.17487/RFC5460, February 2009, <<https://www.rfc-editor.org/info/rfc5460>>.
- [RFC7653] Raghuvanshi, D., Kinnear, K., and D. Kukrety, "DHCPv6 Active Leasequery", RFC 7653, DOI 10.17487/RFC7653, October 2015, <<https://www.rfc-editor.org/info/rfc7653>>.
- [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>>.

- [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

- [BCP38] IETF, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing <https://tools.ietf.org/html/bcp38>", RFC 2827, BCP 38.
- [RFC7283] Cui, Y., Sun, Q., and T. Lemon, "Handling Unknown DHCPv6 Messages", RFC 7283, DOI 10.17487/RFC7283, July 2014, <<https://www.rfc-editor.org/info/rfc7283>>.
- [RFC8213] Volz, B. and Y. Pal, "Security of Messages Exchanged between Servers and Relay Agents", RFC 8213, DOI 10.17487/RFC8213, August 2017, <<https://www.rfc-editor.org/info/rfc8213>>.

## Authors' Addresses

Ian Farrer  
Deutsche Telekom AG  
Landgrabenweg 151  
Bonn, NRW 53227  
DE

Email: [ian.farrer@telekom.de](mailto:ian.farrer@telekom.de)

Naveen Kottapalli  
Benu Networks  
300 Concord Road  
Billerica, MA 01821  
US

Email: [naveen.sarma@gmail.com](mailto:naveen.sarma@gmail.com)

Martin Hunek  
Technical University of Liberec  
Studentska 1402/2  
Liberec, L 46017  
CZ

Email: [martin.hunek@tul.cz](mailto:martin.hunek@tul.cz)

Richard Patterson

Email: richard@helix.net.nz

DHC Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

I. Farrer, Ed.  
Deutsche Telekom AG  
7 March 2022

YANG Data Model for DHCPv6 Configuration  
draft-ietf-dhc-dhcpv6-yang-25

Abstract

This document describes YANG data modules for the configuration and management of DHCPv6 (Dynamic Host Configuration Protocol for IPv6 RFC8415) servers, relays, and clients.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 . . . . .	2
1.1. Scope . . . . .	3
1.2. Extensibility of the DHCPv6 Server YANG Module . . . . .	3
1.2.1. DHCPv6 Option Definitions . . . . .	4
1.3. Terminology . . . . .	6
2. Requirements Language . . . . .	6
3. DHCPv6 Tree Diagrams . . . . .	6
3.1. DHCPv6 Server Tree Diagram . . . . .	6
3.2. DHCPv6 Relay Tree Diagram . . . . .	13
3.3. DHCPv6 Client Tree Diagram . . . . .	16
4. DHCPv6 YANG Modules . . . . .	20
4.1. DHCPv6 Common YANG Module . . . . .	20
4.2. DHCPv6 Server YANG Module . . . . .	29
4.3. DHCPv6 Relay YANG Module . . . . .	50
4.4. DHCPv6 Client YANG Module . . . . .	60
5. Security Considerations . . . . .	75
6. IANA Considerations . . . . .	77
6.1. URI Registration . . . . .	77
6.2. YANG Module Name Registration . . . . .	78
7. Acknowledgments . . . . .	78
8. Contributors . . . . .	78
9. References . . . . .	79
9.1. Normative References . . . . .	79
9.2. Informative References . . . . .	82
Appendix A. Data Tree Examples . . . . .	82
A.1. DHCPv6 Server Configuration Examples . . . . .	82
A.2. DHCPv6 Relay Configuration Example . . . . .	86
A.3. DHCPv6 Client Configuration Example . . . . .	87
Appendix B. Example of Augmenting Additional DHCPv6 Option Definitions . . . . .	90
Appendix C. Example Vendor Specific Server Configuration Module . . . . .	93
Appendix D. Example definition of class-selector configuration . . . . .	99
Author's Address . . . . .	106

## 1. Introduction

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

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

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

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

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

### 1.1. Scope

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

### 1.2. Extensibility of the DHCPv6 Server YANG Module

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

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

In DHCPv6, the concept of 'class selection' for messages received by the server is common. This is the identification and classification of messages based on a number of parameters so that the correct provisioning information can be supplied. For example, allocating a prefix from the correct pool, or supplying a set of options relevant for a specific vendor's client implementation. During the development of this document, implementations were researched and the findings were that while this function is common to all, the method for configuring and implementing this function differs greatly. Therefore, configuration of the class selection function has been omitted from the DHCPv6 server module to allow implementors to define their own suitable YANG modules. Appendix D provides an example of this, to demonstrate how this 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 modeled. For example, OPTION\_IA\_NA (3) is created by the DHCP server when requested by the client. The contents of the fields in the option are based on a number of input configuration parameters which the server will apply when it receives the request (e.g., the T1/T2 timers that are relevant for the pool of addresses). As a result, there are no fields that are directly configurable for the option, so it is not modeled.

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

Name	Server	Relay	Client	Module Name
OPTION_ORO (6) Option Request Option			X	ietf-dhcpv6-client.yang
OPTION_PREFERENCE (7) Preference Option	X			ietf-dhcpv6-server.yang
OPTION_AUTH (11) Authentication Option	X	X		ietf-dhcpv6-common.yang
OPTION_UNICAST (12)	X			ietf-dhcpv6-server.yang

Server Unicast Option				
OPTION_RAPID_COMMIT (14) Rapid Commit Option	X		X	ietf-dhcpv6-common.yang
OPTION_USER_CLASS (15) User Class Option			X	ietf-dhcpv6-client.yang
OPTION_VENDOR_CLASS (16) Vendor Class Option			X	ietf-dhcpv6-client.yang
OPTION_VENDOR_OPTS (17) Vendor-specific Information Option	X		X	ietf-dhcpv6-common.yang
OPTION_INTERFACE_ID (18) Interface-Id Option		X		ietf-dhcpv6-relay.yang
OPTION_RECONF_MSG (19) Reconfigure Message Option	X			ietf-dhcpv6-server.yang
OPTION_RECONF_ACCEPT (20) Reconfigure Accept Option	X		X	ietf-dhcpv6-client.yang
OPTION_INFORMATION _REFRESH_TIME (32) Information Refresh Time Option	X			ietf-dhcpv6-server.yang
OPTION_SOL_MAX_RT (82) sol max rt Option	X			ietf-dhcpv6-server.yang
OPTION_INF_MAX_RT (83) inf max rt Option	X			ietf-dhcpv6-server.yang

Table 1: Modeled DHCPv6 Options



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

### 1.3. Terminology

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

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

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

## 2. 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.

## 3. DHCPv6 Tree Diagrams

### 3.1. 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 defined in Section 4.1.

```
module: ietf-dhcpv6-server
  +--rw dhcpv6-server
    +--rw enabled?                boolean
    +--rw server-duid?            dhc6:duid
    +--rw vendor-config
    +--rw option-sets
      +--rw option-set* [option-set-id]
        +--rw option-set-id      string
        +--rw description?       string
        +--rw preference-option
          +--rw pref-value?      uint8
        +--rw auth-option
          +--rw algorithm?       uint8
```

```

+--rw rdm?                               uint8
+--rw replay-detection?                   uint64
+--rw (protocol)?
  +--:(conf-token)
    | +--rw token-auth-information?       binary
  +--:(rkap)
    +--rw datatype?                       uint8
    +--rw auth-info-value?                 binary
+--rw server-unicast-option
  | +--rw server-address?                  inet:ipv6-address
+--rw rapid-commit-option!
+--rw vendor-specific-information-options
  | +--rw vendor-specific-information-option*
    [enterprise-number]
    +--rw enterprise-number                uint32
    +--rw vendor-option-data* [sub-option-code]
      +--rw sub-option-code                uint16
      +--rw sub-option-data?               binary
+--rw reconfigure-message-option
  | +--rw msg-type?                        uint8
+--rw reconfigure-accept-option!
+--rw info-refresh-time-option
  | +--rw info-refresh-time?               dhc6:timer-seconds32
+--rw sol-max-rt-option
  | +--rw sol-max-rt-value?                dhc6:timer-seconds32
+--rw inf-max-rt-option
  | +--rw inf-max-rt-value?                dhc6:timer-seconds32
+--rw class-selector
+--rw allocation-ranges
  +--rw option-set-id*                    leafref
  +--rw valid-lifetime?                   dhc6:timer-seconds32
  +--rw renew-time?                       dhc6:timer-seconds32
  +--rw rebind-time?                      dhc6:timer-seconds32
  +--rw preferred-lifetime?               dhc6:timer-seconds32
  +--rw rapid-commit?                     boolean
  +--rw allocation-range* [id]
    +--rw id                               string
    +--rw description?                     string
    +--rw network-prefix                   inet:ipv6-prefix
    +--rw option-set-id*                   leafref
    +--rw valid-lifetime?                   dhc6:timer-seconds32
    +--rw renew-time?                       dhc6:timer-seconds32
    +--rw rebind-time?                     dhc6:timer-seconds32
    +--rw preferred-lifetime?               dhc6:timer-seconds32
    +--rw rapid-commit?                     boolean
  +--rw address-pools {na-assignment}?
    | +--rw address-pool* [pool-id]
      +--rw pool-id                        string

```

```

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

```

```

    +--ro lease-t1?
    |   dhc6:timer-seconds32
    +--ro lease-t2?
    |   dhc6:timer-seconds32
    +--ro status
    |   +--ro code?      uint16
    |   +--ro message?   string
+--rw prefix-pools {prefix-delegation}?
+--rw prefix-pool* [pool-id]
+--rw pool-id          string
+--rw pool-prefix
+--rw   inet:ipv6-prefix
+--rw client-prefix-length      uint8
+--rw max-pd-space-utilization? dhc6:threshold
+--rw option-set-id*           leafref
+--rw valid-lifetime?
+--rw   dhc6:timer-seconds32
+--rw renew-time?
+--rw   dhc6:timer-seconds32
+--rw rebind-time?
+--rw   dhc6:timer-seconds32
+--rw preferred-lifetime?
+--rw   dhc6:timer-seconds32
+--rw rapid-commit?           boolean
+--rw host-reservations
+--rw   prefix-reservation* [reserved-prefix]
+--rw     client-duid?        dhc6:duid
+--rw     reserved-prefix
+--rw       inet:ipv6-prefix
+--rw     reserved-prefix-len? uint8
+--rw option-set-id*           leafref
+--rw valid-lifetime?
+--rw   dhc6:timer-seconds32
+--rw renew-time?
+--rw   dhc6:timer-seconds32
+--rw rebind-time?
+--rw   dhc6:timer-seconds32
+--rw preferred-lifetime?
+--rw   dhc6:timer-seconds32
+--rw rapid-commit?           boolean
+--ro active-leases
+--ro total-count      uint64
+--ro allocated-count  uint64
+--ro active-lease* [leased-prefix]
+--ro   leased-prefix
+--ro     inet:ipv6-prefix
+--ro client-duid?      dhc6:duid
+--ro ia-id             uint32

```

```

|         +---ro allocation-time?
|         |         yang:date-and-time
+---ro last-renew-rebind?
|         |         yang:date-and-time
+---ro preferred-lifetime?
|         |         dhc6:timer-seconds32
+---ro valid-lifetime?
|         |         dhc6:timer-seconds32
+---ro lease-t1?
|         |         dhc6:timer-seconds32
+---ro lease-t2?
|         |         dhc6:timer-seconds32
+---ro status
|         +---ro code?          uint16
|         +---ro message?      string
+---rw statistics
  +---rw discontinuity-time?      yang:date-and-time
  +---ro solicit-count?          yang:counter32
  +---ro advertise-count?        yang:counter32
  +---ro request-count?          yang:counter32
  +---ro confirm-count?          yang:counter32
  +---ro renew-count?            yang:counter32
  +---ro rebind-count?           yang:counter32
  +---ro reply-count?            yang:counter32
  +---ro release-count?          yang:counter32
  +---ro decline-count?          yang:counter32
  +---ro reconfigure-count?       yang:counter32
  +---ro information-request-count? yang:counter32
  +---ro discarded-message-count? yang:counter32

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

notifications:
+---n address-pool-utilization-threshold-exceeded
|   {na-assignment}?
|   +---ro pool-id                leafref
|   +---ro total-pool-addresses    uint64
|   +---ro max-allocated-addresses uint64

```

```

|   +---ro allocated-address-count      uint64
+---n prefix-pool-utilization-threshold-exceeded
|   {prefix-delegation}?
|   +---ro pool-id                      leafref
|   +---ro total-pool-prefixes          uint64
|   +---ro max-allocated-prefixes       uint64
|   +---ro allocated-prefixes-count     uint64
+---n invalid-client-detected
|   +---ro message-type?                enumeration
|   +---ro duid?                       dhcp6:duid
|   +---ro description?                 string
+---n decline-received {na-assignment}?
|   +---ro duid?                       dhcp6:duid
|   +---ro declined-resources* []
|       +---ro (resource-type)?
|           +---:(declined-address)
|               | +---ro address?      inet:ipv6-address
|               +---:(declined-prefix)
|                   +---ro prefix?     inet:ipv6-prefix
+---n non-success-code-sent
|   +---ro duid?                       dhcp6:duid
|   +---ro status
|       +---ro code?                   uint16
|       +---ro message?                string

```

Figure 1: DHCPv6 Server Data Module Structure

## Descriptions of important nodes:

- \* **enabled:** Enables/disables the function of the DHCPv6 server.
- \* **dhcpv6-server:** This container holds the server's DHCPv6 specific configuration.
- \* **server-duid:** Each server must have a DUID (DHCP Unique Identifier) to identify itself to clients. A DUID consists of a two-octet type field and an arbitrary length (of no more than 128-octets) content field. Currently there are four DUID types defined in [RFC8415] and [RFC6355]. The DUID may be configured using the format for one of these types, or using the 'unstructured' format. The DUID type definitions are imported from the 'ietf-dhcpv6-common.yang' module. [IANA-HARDWARE-TYPES] and [IANA-PEN] are referenced for the relevant DUID types.
- \* **vendor-config:** This container is provided as a location for additional implementation-specific YANG nodes for the configuration of the device to be augmented. See Appendix C for an example of such a module.

- \* **option-sets:** The server can be configured with multiple option-sets. These are groups of DHCPv6 options with common parameters which will be supplied to clients on request. The 'option-set-id' field is used to reference an option-set elsewhere in the server's configuration.
- \* **option-set:** Holds configuration parameters for DHCPv6 options. The initial set of applicable option definitions are defined here and additional options that are also relevant to the relay and/or client are imported from the 'ietf-dhcpv6-common' module. Where needed, other DHCPv6 option modules can be augmented as they are defined.
- \* **class-selector:** This is provided as a location for additional implementation specific YANG nodes for vendor specific class selector nodes to be augmented. See Appendix D for an example of this.
- \* **allocation-ranges:** A hierarchical model is used for the allocation of addresses and prefixes. The top level 'allocation-ranges' container holds global configuration parameters. Under this, the 'allocation-range' list is used for specifying IPv6 prefixes and additional, prefix specific parameters.
- \* **address-pools:** Used for IA\_NA and IA\_TA pool allocations with a container for defining host reservations. State information about active leases from each pool is also located here.
- \* **prefix-pools:** Defines pools to be used for prefix delegation to clients. Static host reservations can also be configured. As prefix delegation is not supported by all DHCPv6 server implementations, it is enabled by a feature statement.

#### 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.

#### Information about notifications:

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

- \* `invalid-client-detected`: Raised when the server detects an invalid client. A description of the error and message type that has generated the notification can be included.
- \* `decline-received`: Raised when a DHCPv6 Decline message is received from a client.
- \* `non-success-code-sent`: Raised when there is a status message for a failure.

### 3.2. 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 defined in Section 4.1.

The RPCs in the module are taken from requirements defined in [RFC8987].

```

module: ietf-dhcpv6-relay
  +--rw dhcpv6-relay
    +--rw enabled?          boolean
    +--rw relay-if* [if-name]
      +--rw if-name          if:interface-ref
      +--rw enabled?         boolean
      +--rw destination-address*  inet:ipv6-address
      +--rw link-address?       inet:ipv6-address
      +--rw relay-options
        +--rw auth-option
          +--rw algorithm?      uint8
          +--rw rdm?            uint8
          +--rw replay-detection? uint64
          +--rw (protocol)?
            +--:(conf-token)
              +--rw token-auth-information? binary
            +--:(rkap)
              +--rw datatype?    uint8
              +--rw auth-info-value? binary
        +--rw interface-id-option
          +--rw interface-id?    binary
      +--rw statistics
        +--rw discontinuity-time?
          |   yang:date-and-time
        +--ro solicit-received-count?
          |   yang:counter32
        +--ro advertise-sent-count?
          |   yang:counter32
        +--ro request-received-count?

```



```

|         yang:counter32
+--ro confirm-received-count?
|         yang:counter32
+--ro renew-received-count?
|         yang:counter32
+--ro rebind-received-count?
|         yang:counter32
+--ro reply-sent-count?
|         yang:counter32
+--ro release-received-count?
|         yang:counter32
+--ro decline-received-count?
|         yang:counter32
+--ro reconfigure-sent-count?
|         yang:counter32
+--ro information-request-received-count?
|         yang:counter32
+--ro unknown-message-received-count?
|         yang:counter32
+--ro unknown-message-sent-count?
|         yang:counter32
+--ro discarded-message-count?
|         yang:counter32
+--rw prefix-delegation! {prefix-delegation}?
+--ro pd-leases* [ia-pd-prefix]
+--ro ia-pd-prefix          inet:ipv6-prefix
+--ro last-renew?           yang:date-and-time
+--ro client-peer-address?  inet:ipv6-address
+--ro client-duid?          dhc6:duid
+--ro server-duid?          dhc6:duid
+--rw statistics
+--ro relay-forward-sent-count?
|         yang:counter32
+--ro relay-forward-received-count?
|         yang:counter32
+--ro relay-reply-received-count?
|         yang:counter32
+--ro relay-forward-unknown-sent-count?
|         yang:counter32
+--ro relay-forward-unknown-received-count?
|         yang:counter32
+--ro discarded-message-count?
|         yang:counter32

rpcs:
+---x clear-prefix-entry {prefix-delegation}?
| +---w input
| | +---w lease-prefix    leafref

```

```

    | +--ro output
    |   +--ro return-message?  string
+---x clear-client-prefixes {prefix-delegation}?
    | +---w input
    |   | +---w client-duid      dhc6:duid
    |   +--ro output
    |     +--ro return-message?  string
+---x clear-interface-prefixes {prefix-delegation}?
    | +---w input
    |   | +---w interface      -> /dhcpv6-relay/relay-if/if-name
    |   +--ro output
    |     +--ro return-message?  string
notifications:
+---n relay-event
    +--ro topology-change
    +--ro relay-if-name?
    |   -> /dhcpv6-relay/relay-if/if-name
    +--ro last-ipv6-addr?  inet:ipv6-address

```

Figure 2: DHCPv6 Relay Data Module Structure

#### Descriptions of important nodes:

- \* **enabled:** Globally enables/disables all DHCPv6 relay functions.
- \* **dhcpv6-relay:** This container holds the relay's DHCPv6-specific configuration.
- \* **relay-if:** As a relay may have multiple client-facing interfaces, they are configured in a list. The if-name leaf is the key and is an interface-ref to the applicable interface defined by the 'ietf-interfaces' YANG module.
- \* **enabled:** Enables/disables all DHCPv6 relay functions 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 RPCs

- \* **clear-prefix-entry:** 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.

#### Information about notifications:

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

### 3.3. DHCPv6 Client Tree Diagram

The tree diagram in Figure 3 provides an overview of the DHCPv6 client module. The tree also includes the common functions module defined in Section 4.1.

```

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 interface-duid?  dhc6:duid
      |   {(non-temp-addr or prefix-delegation or temp-addr)
      |   and anon-profile)?
      +--rw client-configured-options
      |   +--rw option-request-option
      |   |   +--rw oro-option*  uint16
      |   +--rw rapid-commit-option!
      |   +--rw user-class-option!
      |   |   +--rw user-class-data-instance*

```

```

        [user-class-data-id]
        +--rw user-class-data-id      uint8
        +--rw user-class-data?        binary
    +--rw vendor-class-option
        +--rw vendor-class-option-instances*
            [enterprise-number]
            +--rw enterprise-number      uint32
            +--rw vendor-class-data-element*
                [vendor-class-data-id]
                +--rw vendor-class-data-id      uint8
                +--rw vendor-class-data?        binary
    +--rw vendor-specific-information-options
        +--rw vendor-specific-information-option*
            [enterprise-number]
            +--rw enterprise-number      uint32
            +--rw vendor-option-data* [sub-option-code]
                +--rw sub-option-code      uint16
                +--rw sub-option-data?      binary
    +--rw reconfigure-accept-option!
+--rw ia-na* [ia-id] {non-temp-addr}?
    +--rw ia-id      uint32
    +--rw ia-na-options
    +--ro lease-state
        +--ro ia-na-address?      inet:ipv6-address
        +--ro lease-t1?            dhc6:timer-seconds32
        +--ro lease-t2?            dhc6:timer-seconds32
        +--ro preferred-lifetime?  dhc6:timer-seconds32
        +--ro valid-lifetime?      dhc6:timer-seconds32
        +--ro allocation-time?     yang:date-and-time
        +--ro last-renew-rebind?   yang:date-and-time
        +--ro server-duid?         dhc6:duid
        +--ro status
            +--ro code?            uint16
            +--ro message?         string
+--rw ia-ta* [ia-id] {temp-addr}?
    +--rw ia-id      uint32
    +--rw ia-ta-options
    +--ro lease-state
        +--ro ia-ta-address?      inet:ipv6-address
        +--ro preferred-lifetime?  dhc6:timer-seconds32
        +--ro valid-lifetime?      dhc6:timer-seconds32
        +--ro allocation-time?     yang:date-and-time
        +--ro last-renew-rebind?   yang:date-and-time
        +--ro server-duid?         dhc6:duid
        +--ro status
            +--ro code?            uint16
            +--ro message?         string
+--rw ia-pd* [ia-id] {prefix-delegation}?

```

```

+--rw ia-id                               uint32
+--rw prefix-length-hint?                 uint8
+--rw ia-pd-options
+--ro lease-state
  +--ro ia-pd-prefix?                     inet:ipv6-prefix
  +--ro lease-t1?                         dhc6:timer-seconds32
  +--ro lease-t2?                         dhc6:timer-seconds32
  +--ro preferred-lifetime?               dhc6:timer-seconds32
  +--ro valid-lifetime?                   dhc6:timer-seconds32
  +--ro allocation-time?                   yang:date-and-time
  +--ro last-renew-rebind?                 yang:date-and-time
  +--ro server-duid?                       dhc6:duid
  +--ro status
    +--ro code?                           uint16
    +--ro message?                         string
+--rw statistics
  +--rw discontinuity-time?                 yang:date-and-time
  +--ro solicit-count?                     yang:counter32
  +--ro advertise-count?                   yang:counter32
  +--ro request-count?                     yang:counter32
  +--ro confirm-count?                     yang:counter32
  +--ro renew-count?                       yang:counter32
  +--ro rebind-count?                      yang:counter32
  +--ro reply-count?                       yang:counter32
  +--ro release-count?                     yang:counter32
  +--ro decline-count?                     yang:counter32
  +--ro reconfigure-count?                 yang:counter32
  +--ro information-request-count?         yang:counter32
  +--ro discarded-message-count?           yang:counter32

notifications:
+---n invalid-ia-address-detected
  {non-temp-addr or temp-addr}?
  +--ro ia-id                               uint32
  +--ro ia-na-t1-timer?                     uint32
  +--ro ia-na-t2-timer?                     uint32
  +--ro invalid-address?                     inet:ipv6-address
  +--ro preferred-lifetime?                 uint32
  +--ro valid-lifetime?                     uint32
  +--ro ia-options?                         binary
  +--ro description?                       string
+---n transmission-failed
  +--ro failure-type                       enumeration
  +--ro description?                       string
+---n unsuccessful-status-code
  +--ro server-duid                         dhc6:duid
  +--ro status
    +--ro code?                             uint16

```

```

|      +--ro message?   string
+---n server-duid-changed
      {non-temp-addr or prefix-delegation or temp-addr}?
+--ro new-server-duid      dhc6:duid
+--ro previous-server-duid  dhc6:duid
+--ro lease-ia-na?
|      -> /dhcpv6-client/client-if/ia-na/ia-id
|      {non-temp-addr}?
+--ro lease-ia-ta?
|      -> /dhcpv6-client/client-if/ia-ta/ia-id
|      {temp-addr}?
+--ro lease-ia-pd?
|      -> /dhcpv6-client/client-if/ia-pd/ia-id
|      {prefix-delegation}?

```

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/interface-duid`: The DUID (DHCP Unique Identifier) is used to identify the client to servers and relays. A DUID consists of a two-octet type field and an arbitrary length (1-128 octets) content field. Currently there are four DUID types defined in [RFC8415] and [RFC6355]. The DUID may be configured using the format for one of these types, or using the 'unstructured' format. The DUID type definitions are imported from the 'ietf-dhcpv6-common.yang' module. [IANA-HARDWARE-TYPES] and [IANA-PEN] are referenced for the relevant DUID types. A DUID only needs to be configured if the client is requesting addresses and/or prefixes from the server. Presence of the 'client-duid' or 'interface-duid' leaves is conditional on at least one of the 'non-temp-addr', 'temp-addr', or 'prefix-delegation' features being enabled. Additionally, if the 'anon-profile' [RFC7844] feature is enabled, a unique DUID can be configured per DHCP enabled interface using the 'interface-duid' leaf, otherwise there is a global 'client-duid' leaf.
- \* `client-configured-options`: Holds configuration parameters for DHCPv6 options which can be sent by the client. The initial set of applicable option definitions are defined here and additional options that are also relevant to the relay and/or server are imported from the 'ietf-dhcpv6-common' module. Where needed, other DHCPv6 option modules can be augmented as they are defined.
- \* `ia-na, ia-ta, ia-pd`: Contains configuration nodes relevant for requesting one or more of each of the lease types. Read-only nodes related to the active leases for each type are also located here. As these lease types may not be supported by all DHCPv6 client implementations, they are enabled via individual feature statements. Stateless DHCP ([RFC8415] Section 6.1) is configured when all address and prefix features are disabled.

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.

#### 4. DHCPv6 YANG Modules

##### 4.1. DHCPv6 Common YANG Module

This module imports typedefs from [RFC6991].

```
<CODE BEGINS> file "ietf-dhcpv6-common@2022-03-07.yang"

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

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

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

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

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

    Copyright (c) 2022 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 Revised BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX
    (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
    for full legal notices.";

  revision 2022-03-07 {
    description
      "Initial Revision.";
```



```
    reference
      "XXXX: YANG Data Model for DHCPv6 Configuration";
  }

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

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

  typedef duid-base {
    type string {
      pattern '([0-9a-fA-F]{2}){3,130}';
    }
    description
      "Each DHCP server and client has a DUID (DHCP Unique
      Identifier). The DUID consists of a two-octet type field
      and an arbitrary length (1-128 octets) content field.
      The duid-base type is used by other duid types with
      additional pattern constraints.

      Currently, there are four defined types of DUIDs
      in RFC 8415 and RFC 6355 - DUID-LLT, DUID-EN, DUID-LL and
      DUID-UUID. DUID-unstructured represents DUIDs which do not
      follow any of the defined formats.

      Type 'string' is used to represent the hexadecimal DUID value
      so that pattern constraints can be applied.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6), Section 11
      RFC 6355: Definition of the UUID-Based DHCPv6 Unique
      Identifier (DUID-UUID), Section 4";
  }

  typedef duid-llt {
    type duid-base {
      pattern '0001'
        + '[0-9a-fA-F]{12,}';
    }
  }
```

## description

"DUID type 1, based on Link-Layer Address Plus Time (DUID-LLT). Constructed with a 2-octet hardware type assigned by IANA, 4-octets containing the time the DUID is generated (represented in seconds since midnight (UTC), January 1, 2000, modulo  $2^{32}$ ), and a link-layer address. The address is encoded without separator characters. For example:

```
+-----+-----+-----+-----+
| 0001 | 0006 | 28490058 | 00005E005300 |
+-----+-----+-----+-----+
```

This example includes the 2-octet DUID type of 1 (0x01), the hardware type is 0x06 (IEEE Hardware Types) the creation time is 0x28490058 (constructed as described above). Finally, the link-layer address is 0x5E005300 (EUI-48 address 00-00-5E-00-53-00)";

reference "RFC 8415: Dynamic Host Configuration Protocol for IPv6 (DHCPv6), Section 11.2  
IANA 'Hardware Types' registry.

<<https://www.iana.org/assignments/arp-parameters>>;

}

## typedef duid-en {

```
type duid-base {
  pattern '0002'
  + '[0-9a-fA-F]{8,}';
}
```

## description

"DUID type 2, assigned by vendor based on Enterprise Number (DUID-EN). This DUID consists of the 4-octet vendor's registered Private Enterprise Number as maintained by IANA followed by a unique identifier assigned by the vendor. For example:

```
+-----+-----+-----+-----+
| 0002 | 00007ED9 | 0CC084D303000912 |
+-----+-----+-----+-----+
```

This example includes the 2-octet DUID type of 2 (0x02), 4-octets for the Enterprise Number (0x7ED9), followed by 8-octets of identifier data (0x0CC084D303000912).";

reference "RFC 8415: Dynamic Host Configuration Protocol for IPv6 (DHCPv6), Section 11.3

IANA 'Private Enterprise Numbers' registry.

<<https://www.iana.org/assignments/enterprise-numbers>>;

}

```

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

        +-----+-----+-----+
        | 0003 | 0006 | 00005E005300 |
        +-----+-----+-----+

        This example includes the 2-octet DUID type of 3 (0x03), the
        hardware type is 0x06 (IEEE Hardware Types), and the
        link-layer address is 0x5E005300 (EUI-48 address
        00-00-5E-00-53-00)";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11.4
        IANA 'Hardware Types' registry.
        <https://www.iana.org/assignments/arp-parameters>";
}

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

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

        This example includes the 2-octet DUID type of 4 (0x04), and
        the UUID 9f03b182-7057-47e3-8a1e-422910078642.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11.5
        RFC 6355: Definition of the UUID-Based DHCPv6 Unique
        Identifier (DUID-UUID)";
}

typedef duid-unstructured {

```

```

    type duid-base {
        pattern '(000[1-4].*)' {
            modifier invert-match;
        }
    }
    description
        "Used for DUIDs following any other formats than DUID
        types 1-4. For example:

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

        Here, an arbitrary 16-octet value is used. The only constraint
        placed on this is that the first 2-octets are not 0x01-0x04
        to avoid collision with the other defined DUID types
        (duid-llt, duid-en, duid-ll, or duid-uuid).";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
        IPv6 (DHCPv6), Section 11";
    }

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

    /*
     * Groupings
     */

    grouping status {
        description
            "Holds information about the most recent status code which
            has been sent by the server or received by the client.";
        reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 7.5.";
        container status {
            description

```

```
    "Status code information, relating to the success or failure
    of operations requested in messages.";
  leaf 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 message {
    type string;
    description
      "A UTF-8 encoded text string suitable for display to an
      end user. It MUST NOT be null-terminated.";
  }
}

grouping auth-option-group {
  description
    "OPTION_AUTH (11) Authentication Option.";
  reference "RFC 8415: Dynamic Host Configuration Protocol
  for IPv6 (DHCPv6), Section 21.11
  RFC 3118: Authentication for DHCP Messages
  IANA 'Dynamic Host Configuration Protocol (DHCP)
  Authentication Option Name Spaces' registry.
  <https://www.iana.org/assignments/auth-namespaces>";
  container auth-option {
    description
      "OPTION_AUTH (11) Authentication Option.";
    leaf algorithm {
      type uint8;
      description
        "The algorithm used in the authentication protocol.";
    }
    leaf rdm {
      type uint8;
      description
        "The Replay Detection Method (RDM) used in this
        Authentication option.";
    }
    leaf replay-detection {
      type uint64;
      description
        "The replay detection information for the RDM.";
    }
    choice protocol {
```

```
description
  "The authentication protocol used in the option. Namespace
  values 1 (delayed authentication) and 2 (Delayed
  Authentication (Obsolete) are not applicable and so are
  not modeled.";
case conf-token {
  leaf token-auth-information {
    type binary;
    description
      "Protocol Namespace Value 0. The authentication
      information, as specified by the protocol and
      algorithm used in this Authentication option.";
  }
}
case rkap {
  description
    "Protocol Namespace Value 3. RKAP provides protection
    against misconfiguration of a client caused by a
    Reconfigure message sent by a malicious DHCP server.";
  leaf datatype {
    type uint8 {
      range "1 .. 2";
    }
    description
      "Type of data in the Value field carried in this
      option.
      1 Reconfigure key value (used in the Reply
      message).
      2 HMAC-MD5 digest of the message (used in
      the Reconfigure message).";
  }
  leaf auth-info-value {
    type binary {
      length 16;
    }
    description
      "Data as defined by the Type field. A 16-octet field.";
  }
}
}
}

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

```
    container rapid-commit-option {
      presence "Enable sending of this option";
      description
        "OPTION_RAPID_COMMIT (14) Rapid Commit Option.";
    }
  }

  grouping vendor-specific-information-option-group {
    description
      "OPTION_VENDOR_OPTS (17) Vendor-specific Information
      Option.";
    reference "RFC 8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6), Section 21.17";
    container vendor-specific-information-options {
      description
        "OPTION_VENDOR_OPTS (17) Vendor-specific Information
        Option.";
      list vendor-specific-information-option {
        key enterprise-number;
        description
          "The Vendor-specific Information option allows for
          multiple instances in a single message. Each list entry
          defines the contents of an instance of the option.";
        leaf enterprise-number {
          type uint32;
          description
            "The vendor's registered Enterprise Number, as
            maintained by IANA.";
          reference "IANA 'Private Enterprise Numbers' registry.
          <https://www.iana.org/assignments/enterprise-numbers>";
        }
        list vendor-option-data {
          key sub-option-code;
          description
            "Vendor options, interpreted by vendor-specific
            client/server functions.";
          leaf sub-option-code {
            type uint16;
            description
              "The code for the sub-option.";
          }
          leaf sub-option-data {
            type binary;
            description
              "The data area for the sub-option.";
          }
        }
      }
    }
  }
```

```

    }
  }

  grouping reconfigure-accept-option-group {
    description
      "OPTION_RECONF_ACCEPT (20) Reconfigure Accept Option.
      A client uses the Reconfigure Accept option to announce to
      the server whether the client is willing to accept Reconfigure
      messages, and a server uses this option to tell the client
      whether or not to accept Reconfigure messages. In the absence
      of this option, the default behavior is that the client is
      unwilling to accept Reconfigure messages. The presence node
      is used to enable the option.";
    reference "RFC 8415: Dynamic Host Configuration Protocol
    for IPv6 (DHCPv6), Section 21.20";
    container reconfigure-accept-option {
      presence "Enable sending of this option";
      description
        "OPTION_RECONF_ACCEPT (20) Reconfigure Accept Option.";
    }
  }
}
<CODE ENDS>

```

#### 4.2. DHCPv6 Server YANG Module

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

```

<CODE BEGINS> file "ietf-dhcpv6-server@2022-03-07.yang"

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

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

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

  import ietf-dhcpv6-common {

```



```
    prefix dhc6;
    reference
      "RFC XXXX: To be updated on publication";
  }

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

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

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

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

    Copyright (c) 2022 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 Revised BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions
    Relating to IETF Documents
    (https://trustee.ietf.org/license-info).

    This version of this YANG module is part of RFC XXXX
    (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
    for full legal notices.";

  revision 2022-03-07 {
    description
      "Initial Revision.";
    reference
      "XXXX: YANG Data Model for DHCPv6 Configuration";
  }
```

```
/*
 * Features
 */

feature na-assignment {
  description
    "Denotes that the server implements DHCPv6 non-temporary
    address assignment.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 6.2";
}

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

/*
 * Groupings
 */

grouping resource-config {
  description
    "Nodes that are reused at multiple levels in the DHCPv6
    server's addressing hierarchy.";
  leaf-list option-set-id {
    type leafref {
      path "/dhcpv6-server/option-sets/option-set/option-set-id";
    }
    description
      "The ID field of relevant set of DHCPv6 options (option-set)
      to be provisioned to clients using the allocation-range.";
  }
  leaf valid-lifetime {
    type dhc6:timer-seconds32;
    description
      "Valid lifetime for the Identity Association (IA).";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6), Section 12.1";
  }
  leaf renew-time {
    type dhc6:timer-seconds32;
    description
      "Renew (T1) time.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
```

```
        IPv6 (DHCPv6), Section 4.2";
    }
    leaf rebind-time {
        type dhc6:timer-seconds32;
        description
            "Rebind (T2) time.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 4.2";
    }
    leaf preferred-lifetime {
        type dhc6:timer-seconds32;
        description
            "Preferred lifetime for the Identity Association (IA).";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 12.1";
    }
    leaf rapid-commit {
        type boolean;
        description
            "When set to 'true', Specifies that client-server exchanges
            involving two messages is supported.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 5.1";
    }
}

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

```
        "Time and date that the lease was made.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 18";
    }
    leaf last-renew-rebind {
        type yang:date-and-time;
        description
            "Time of the last successful renew or rebind.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 18";
    }
    leaf preferred-lifetime {
        type dhc6:timer-seconds32;
        description
            "The preferred lifetime expressed in seconds.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 6";
    }
    leaf valid-lifetime {
        type dhc6:timer-seconds32;
        description
            "The valid lifetime for the lease expressed in seconds.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 6";
    }
    leaf lease-t1 {
        type dhc6:timer-seconds32;
        description
            "The time interval after which the client should contact
            the server from which the addresses in the IA_NA were
            obtained to extend the lifetimes of the addresses assigned
            to the IA_PD.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 4.2";
    }
    leaf lease-t2 {
        type dhc6:timer-seconds32;
        description
            "The time interval after which the client should contact
            any available server to extend the lifetimes of the
            addresses assigned to the IA_PD.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 4.2";
    }
    uses dhc6:status;
}

grouping message-statistics {
```

```
description
  "Counters for DHCPv6 messages.";
leaf discontinuity-time {
  type yang:date-and-time;
  description
    "The time on the most recent occasion at which any one or
    more of DHCPv6 server's counters suffered a discontinuity.
    If no such discontinuities have occurred since the last
    re-initialization of the local management subsystem, then
    this node contains the time the local management subsystem
    re-initialized itself.";
}
leaf solicit-count {
  type yang:counter32;
  config "false";
  description
    "Number of Solicit (1) messages received.";
}
leaf advertise-count {
  type yang:counter32;
  config "false";
  description
    "Number of Advertise (2) messages sent.";
}
leaf request-count {
  type yang:counter32;
  config "false";
  description
    "Number of Request (3) messages received.";
}
leaf confirm-count {
  type yang:counter32;
  config "false";
  description
    "Number of Confirm (4) messages received.";
}
leaf renew-count {
  type yang:counter32;
  config "false";
  description
    "Number of Renew (5) messages received.";
}
leaf rebind-count {
  type yang:counter32;
  config "false";
  description
    "Number of Rebind (6) messages received.";
}
```

```
leaf reply-count {
  type yang:counter32;
  config "false";
  description
    "Number of Reply (7) messages sent.";
}
leaf release-count {
  type yang:counter32;
  config "false";
  description
    "Number of Release (8) messages received.";
}
leaf decline-count {
  type yang:counter32;
  config "false";
  description
    "Number of Decline (9) messages received.";
}
leaf reconfigure-count {
  type yang:counter32;
  config "false";
  description
    "Number of Reconfigure (10) messages sent.";
}
leaf information-request-count {
  type yang:counter32;
  config "false";
  description
    "Number of Information-request (11) messages
    received.";
}
leaf discarded-message-count {
  type yang:counter32;
  config "false";
  description
    "Number of messages that have been discarded for any
    reason.";
}
}

grouping preference-option-group {
  description
    "OPTION_PREFERENCE (7) Preference Option.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 21.8";
  container preference-option {
    description
      "OPTION_PREFERENCE (7) Preference Option.";
  }
}
```

```
    leaf pref-value {
      type uint8;
      description
        "The preference value for the server in this message. A
        1-octet unsigned integer.";
    }
  }
}

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

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

grouping info-refresh-time-option-group {
  description
    "OPTION_INFORMATION_REFRESH_TIME (32) Information Refresh
    Time Option.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
```

```
    IPv6 (DHCPv6), Section 21.23";
  container info-refresh-time-option {
    description
      "OPTION_INFORMATION_REFRESH_TIME (32) Information Refresh
      Time Option.";
    leaf info-refresh-time {
      type dhc6:timer-seconds32;
      description
        "Time duration specifying an upper bound for how long a
        client should wait before refreshing information retrieved
        from a DHCP server.";
    }
  }
}

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

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



```
/*
 * Data Nodes
 */

container dhcpv6-server {
  description
    "Configuration nodes for the DHCPv6 server.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 18.3";
  leaf enabled {
    type boolean;
    description
      "Enables the DHCP server function.";
  }
  leaf server-duid {
    type dhc6:duid;
    description
      "DUID of the server.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6), Section 11";
  }
  container vendor-config {
    description
      "This container provides a location for augmenting vendor
        or implementation specific configuration nodes.";
  }
  container option-sets {
    description
      "A server may allow different option sets to be configured
        for clients matching specific parameters such as topological
        location or client type. The 'option-set' list is a set of
        options and their contents that will be returned to
        clients.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
      IPv6 (DHCPv6), Section 21";
    list option-set {
      key option-set-id;
      description
        "YANG definitions for DHCPv6 options are contained in
          separate YANG modules and augmented to this container as
          required.";
      leaf option-set-id {
        type string;
        description
          "Option set identifier.";
      }
      leaf description {
        type string;
      }
    }
  }
}
```

```
        description
            "An optional field for storing additional information
            relevant to the option set.";
    }
    uses preference-option-group;
    uses dhc6:auth-option-group;
    uses server-unicast-option-group;
    uses dhc6:rapid-commit-option-group;
    uses dhc6:vendor-specific-information-option-group;
    uses reconfigure-message-option-group;
    uses dhc6:reconfigure-accept-option-group;
    uses info-refresh-time-option-group;
    uses sol-max-rt-option-group;
    uses inf-max-rt-option-group;
}

container class-selector {
    description
        "DHCPv6 servers use a 'class-selector' function in order
        to identify and classify incoming client messages
        so that they can be given the correct configuration.
        The mechanisms used for implementing this function vary
        greatly between different implementations such it is not
        possible to include in this module. This container provides
        a location for server implementors to augment their own
        class-selector YANG.";
}

container allocation-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 allocation-ranges. Under
        this are the individual allocation-ranges.";
    uses resource-config;
    list allocation-range {
        key id;
        description
            "Network-ranges are identified by the 'id' key.";
        leaf id {
            type string;
            mandatory true;
            description
                "Unique identifier for the allocation range.";
        }
        leaf description {
            type string;
        }
    }
}
```

```
    description
      "Description for the allocation range.";
  }
  leaf network-prefix {
    type inet:ipv6-prefix;
    mandatory true;
    description
      "Network prefix.";
  }
  uses resource-config;
  container address-pools {
    if-feature na-assignment;
    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 string;
        mandatory true;
        description
          "Unique identifier for the pool.";
      }
      leaf pool-prefix {
        type inet:ipv6-prefix;
        mandatory true;
        description
          "IPv6 prefix for the pool. Should be contained
          within the network-prefix, if configured.";
      }
      leaf start-address {
        type inet:ipv6-address-no-zone;
        mandatory true;
        description
          "Starting IPv6 address for the pool.";
      }
      leaf end-address {
        type inet:ipv6-address-no-zone;
        mandatory true;
        description
          "Ending IPv6 address for the pool.";
      }
      leaf max-address-utilization {
        type dhc6:threshold;
        description
```

```
    "Maximum amount of the addresses in the
    pool which can be simultaneously allocated,
    calculated as a percentage of the available
    addresses (end-address minus start-address plus
    one), rounded up. Used to set the value for the
    address-pool-utilization-threshold-exceeded
    notification";
  }
  uses resource-config;
  container host-reservations {
    description
      "Configuration for host reservations from the
      address pool.";
    list host-reservation {
      key reserved-addr;
      description
        "List of host reservations.";
      leaf client-duid {
        type dhc6:duid;
        description
          "Client DUID for the reservation.";
      }
      leaf reserved-addr {
        type inet:ipv6-address;
        description
          "Reserved IPv6 address.";
      }
    }
    uses resource-config;
  }
}
container active-leases {
  config false;
  description
    "Holds state related to active client
    leases.";
  leaf total-count {
    type uint64;
    mandatory true;
    description
      "The total number of addresses in the pool.";
  }
  leaf allocated-count {
    type uint64;
    mandatory true;
    description
      "The number of addresses or prefixes in the pool
      that are currently allocated.";
  }
}
```

```
    list active-lease {
      key leased-address;
      description
        "List of active address leases.";
      leaf leased-address {
        type inet:ipv6-address;
        description
          "Active address lease entry.";
      }
      uses lease-information;
    }
  }
}

container prefix-pools {
  if-feature prefix-delegation;
  description
    "Configuration for the DHCPv6 server's prefix pools.";
  list prefix-pool {
    key pool-id;
    description
      "List of prefix pools for allocation to clients,
        distinguished by 'pool-id'.";
    leaf pool-id {
      type string;
      mandatory true;
      description
        "Unique identifier for the pool.";
    }
    leaf pool-prefix {
      type inet:ipv6-prefix;
      mandatory true;
      description
        "IPv6 prefix for the pool. Should be contained
          within the network-prefix, if configured.";
    }
    leaf client-prefix-length {
      type uint8 {
        range "1 .. 128";
      }
      mandatory true;
      description
        "Length of the prefixes that will be delegated
          to clients.";
    }
    leaf max-pd-space-utilization {
      type dhc6:threshold;
      description
```

```
    "Maximum amount of the prefixes in the pool which
    can be simultaneously allocated, calculated as a
    percentage of the available prefixes, rounded up.
    Used to set the value for the
    prefix-pool-utilization-threshold-exceeded
    notification";
}
uses resource-config;
container host-reservations {
  description
    "Configuration for host reservations from the
    prefix pool.";
  list prefix-reservation {
    key reserved-prefix;
    description
      "Reserved prefix reservation.";
    leaf client-duid {
      type dhc6:duid;
      description
        "Client DUID for the reservation.";
    }
    leaf reserved-prefix {
      type inet:ipv6-prefix;
      description
        "Reserved IPv6 prefix";
    }
    leaf reserved-prefix-len {
      type uint8;
      description
        "Reserved IPv6 prefix length.";
    }
  }
}
uses resource-config;
}
container active-leases {
  config false;
  description
    "Holds state related to active client prefix
    leases.";
  leaf total-count {
    type uint64;
    mandatory true;
    description
      "The total number of prefixes in the pool.";
  }
  leaf allocated-count {
    type uint64;
    mandatory true;
```

```

        description
            "The number of prefixes in the pool that are
            currently allocated.";
    }
    list active-lease {
        key leased-prefix;
        description
            "List of active prefix leases.";
        leaf leased-prefix {
            type inet:ipv6-prefix;
            description
                "Active leased prefix entry.";
        }
        uses lease-information;
    }
}
}
}
}
}
container statistics {
    description
        "DHCPv6 message counters for the server.";
    uses message-statistics;
}
}
}
}
}
/*
 * RPCs
 */

rpc delete-address-lease {
    nacm:default-deny-all;
    if-feature na-assignment;
    description
        "Deletes a client's active address lease from the server's
        lease database. Note this will not cause the address to be
        revoked from the client, and the lease may be refreshed or
        renewed by the client.";
    input {
        leaf lease-address-to-delete {
            type leafref {
                path "/dhcpv6-server/allocation-ranges/" +
                    "allocation-range/address-pools/address-pool" +
                    "/active-leases/active-lease/leased-address";
            }
            mandatory true;
            description

```

```
        "IPv6 address of an active lease that will be
        deleted from the server.";
    }
}
output {
    leaf return-message {
        type string;
        description
            "Response message from the server. If available, a
            language identifier should be included in the message.";
        reference "BCP 14 (RFC 2277) IETF Policy on Character Sets
            and Languages, Section 4.2.";
    }
}
}

rpc delete-prefix-lease {
    nacm:default-deny-all;
    if-feature prefix-delegation;
    description
        "Deletes a client's active prefix lease from the server's
        lease database. Note, this will not cause the prefix to be
        revoked from the client, and the lease may be refreshed or
        renewed by the client.";
    input {
        leaf lease-prefix-to-delete {
            type leafref {
                path "/dhcpv6-server/allocation-ranges/" +
                    "allocation-range/prefix-pools/prefix-pool" +
                    "/active-leases/active-lease/leased-prefix";
            }
            mandatory true;
            description
                "IPv6 prefix of an active lease that will be deleted
                from the server.";
        }
    }
    output {
        leaf return-message {
            type string;
            description
                "Response message from the server. If available, a
                language identifier should be included in the message.";
            reference "BCP 14 (RFC 2277) IETF Policy on Character Sets
                and Languages, Section 4.2.";
        }
    }
}
}
```



```
/*
 * Notifications
 */

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

notification prefix-pool-utilization-threshold-exceeded {
  if-feature prefix-delegation;
```

```
description
  "Notification sent when the prefix pool utilization
  exceeds the threshold configured in
  max-pd-space-utilization.";
leaf pool-id {
  type leafref {
    path "/dhcpv6-server/allocation-ranges" +
    "/allocation-range/prefix-pools/prefix-pool/pool-id";
  }
  mandatory true;
  description
    "Unique identifier for the pool.";
}
leaf total-pool-prefixes {
  type uint64;
  mandatory true;
  description
    "Total number of prefixes in the pool.";
}
leaf max-allocated-prefixes {
  type uint64;
  mandatory true;
  description
    "Maximum number of prefixes that can be simultaneously
    allocated from the pool. This value may be less than
    count of total prefixes. Calculated as the
    max-prefix-utilization (percentage) of the
    total-pool-prefixes, rounded up.";
}
leaf allocated-prefixes-count {
  type uint64;
  mandatory true;
  description
    "Number of prefixes allocated from the pool.";
}
}

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

```
        description
            "Request (3) message.";
    }
    enum confirm {
        description
            "Confirm (4) message.";
    }
    enum renew {
        description
            "Renew (5) message.";
    }
    enum rebind {
        description
            "Rebind (6) message.";
    }
    enum release {
        description
            "Release (8) message.";
    }
    enum decline {
        description
            "Decline (9) message.";
    }
    enum info-request {
        description
            "Information request (11) message.";
    }
}
description
    "The message type received by the server that has caused
    the error.";
}
leaf duid {
    type dhc6:duid;
    description
        "Client DUID.";
}
leaf description {
    type string;
    description
        "Description of the event (e.g., an error code or log
        message).";
}
}

notification decline-received {
    if-feature na-assignment;
    description
```

```
    "Notification sent when the server has received a Decline (9)
    message from a client.";
  leaf duid {
    type dhc6:duid;
    description
      "Client DUID.";
  }
  list declined-resources {
    description
      "List of declined addresses and/or prefixes.";
    choice resource-type {
      description
        "Type of resource that has been declined.";
      case declined-address {
        leaf address {
          type inet:ipv6-address;
          description
            "Address that has been declined.";
        }
      }
      case declined-prefix {
        leaf prefix {
          type inet:ipv6-prefix;
          description
            "Prefix that has been declined.";
        }
      }
    }
  }
}

notification non-success-code-sent {
  description
    "Notification sent when the server responded to a client with
    a non-success status code.";
  leaf duid {
    type dhc6:duid;
    description
      "Client DUID.";
  }
  uses dhc6:status;
}
}
<CODE ENDS>
```

## 4.3. DHCPv6 Relay YANG Module

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

<CODE BEGINS> file "ietf-dhcpv6-relay@2022-03-07.yang"

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

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

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

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

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

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

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

  contact
    "WG Web:  <https://datatracker.ietf.org/wg/dhc/>
    WG List:  <mailto:dhcwg@ietf.org>
    Author:   Yong Cui <yong@csnet1.cs.tsinghua.edu.cn>
```

Author: Linhui Sun <lh.sunlinh@gmail.com>  
Editor: Ian Farrer <ian.farrer@telekom.de>  
Author: Sladjana Zeichlin <sladjana.zechlin@telekom.de>  
Author: Zihao He <hezihao9512@gmail.com>  
Author: Michal Nowikowski <godfryd@isc.org>;

description

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

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

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

```
revision 2022-03-07 {  
  description  
    "Initial Revision.";  
  reference  
    "XXXX: YANG Data Model for DHCPv6 Configuration";  
}
```

```
/*  
 * Features  
 */
```

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

```
    }

    /*
     * Groupings
     */

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

    grouping message-statistics {
        description
            "Contains counters for the different DHCPv6 message types.";
        leaf discontinuity-time {
            type yang:date-and-time;
```

```
description
  "The time on the most recent occasion at which any one or
  more of DHCPv6 relay's counters suffered a discontinuity.
  If no such discontinuities have occurred since the last
  re-initialization of the local management subsystem, then
  this node contains the time the local management subsystem
  re-initialized itself.";
}
leaf solicit-received-count {
  type yang:counter32;
  config "false";
  description
    "Number of Solicit (1) messages received.";
}
leaf advertise-sent-count {
  type yang:counter32;
  config "false";
  description
    "Number of Advertise (2) messages sent.";
}
leaf request-received-count {
  type yang:counter32;
  config "false";
  description
    "Number of Request (3) messages received.";
}
leaf confirm-received-count {
  type yang:counter32;
  config "false";
  description
    "Number of Confirm (4) messages received.";
}
leaf renew-received-count {
  type yang:counter32;
  config "false";
  description
    "Number of Renew (5) messages received.";
}
leaf rebind-received-count {
  type yang:counter32;
  config "false";
  description
    "Number of Rebind (6) messages received.";
}
leaf reply-sent-count {
  type yang:counter32;
  config "false";
  description
```



```
        "Number of Reply (7) messages sent.";
    }
    leaf release-received-count {
        type yang:counter32;
        config "false";
        description
            "Number of Release (8) messages received.";
    }
    leaf decline-received-count {
        type yang:counter32;
        config "false";
        description
            "Number of Decline (9) messages received.";
    }
    leaf reconfigure-sent-count {
        type yang:counter32;
        config "false";
        description
            "Number of Reconfigure (10) messages sent.";
    }
    leaf information-request-received-count {
        type yang:counter32;
        config "false";
        description
            "Number of Information-request (11) messages
            received.";
    }
    leaf unknown-message-received-count {
        type yang:counter32;
        config "false";
        description
            "Number of messages of unknown type that have
            been received.";
    }
    leaf unknown-message-sent-count {
        type yang:counter32;
        config "false";
        description
            "Number of messages of unknown type that have
            been sent.";
    }
    leaf discarded-message-count {
        type yang:counter32;
        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 yang:counter32;
    config "false";
    description
      "Number of Relay-forward (12) messages sent.";
  }
  leaf relay-forward-received-count {
    type yang:counter32;
    config "false";
    description
      "Number of Relay-forward (12) messages received.";
  }
  leaf relay-reply-received-count {
    type yang:counter32;
    config "false";
    description
      "Number of Relay-reply (13) messages received.";
  }
  leaf relay-forward-unknown-sent-count {
    type yang:counter32;
    config "false";
    description
      "Number of Relay-forward (12) messages containing
      a message of unknown type sent.";
  }
  leaf relay-forward-unknown-received-count {
    type yang:counter32;
    config "false";
    description
      "Number of Relay-forward (12) messages containing
      a message of unknown type received.";
  }
  leaf discarded-message-count {
    type yang:counter32;
    config "false";
    description
      "Number of messages that have been discarded
      for any reason.";
  }
}

grouping interface-id-option-group {
  description
```

```
    "OPTION_INTERFACE_ID (18) Interface-Id Option.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 21.18";
  container interface-id-option {
    description
      "OPTION_INTERFACE_ID (18) Interface-Id Option.";
    leaf interface-id {
      type binary;
      description
        "An opaque value of arbitrary length generated by the
        relay agent to identify one of the relay agent's
        interfaces.";
    }
  }
}

/*
 * Data Nodes
 */

container dhcpv6-relay {
  description
    "This container contains the configuration data nodes
    for the relay.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 19";
  leaf enabled {
    type boolean;
    description
      "Globally enables the DHCP relay function.";
  }
  list relay-if {
    key if-name;
    description
      "List of interfaces configured for DHCPv6 relaying.";
    leaf if-name {
      type if:interface-ref;
      description
        "interface-ref to the relay interface.";
    }
    leaf enabled {
      type boolean;
      description
        "Enables the DHCP relay function for this interface.";
    }
  }
  leaf-list destination-address {
    type inet:ipv6-address;
    description

```

```
        "Each DHCPv6 relay agent may be configured with a list
        of destination addresses for relayed messages.
        The list may include unicast addresses, multicast
        addresses or other valid addresses.";
    }
    leaf link-address {
        type inet:ipv6-address;
        description
            "An address that may be used by the server to identify
            the link on which the client is located.";
    }
    container relay-options {
        description
            "Definitions for DHCPv6 options that can be sent
            by the relay are augmented to this location from other
            YANG modules as required.";
        uses dhc6:auth-option-group;
        uses interface-id-option-group;
    }
    container statistics {
        description
            "DHCPv6 message counters for the relay's interface.";
        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;
    }
}

container statistics {
    description
        "Global DHCPv6 message counters for the relay.";
    uses global-statistics;
}

/*
 * RPCs
 */

rpc clear-prefix-entry {
    nacm:default-deny-all;
    if-feature prefix-delegation;
    description
```

```
    "Clears an entry for an active delegated prefix
    from the relay.";
reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
Section 4.4";
input {
  leaf lease-prefix {
    type leafref {
      path "/dhcpv6-relay/relay-if/prefix-delegation" +
        "/pd-leases/ia-pd-prefix";
    }
    mandatory true;
    description
      "IPv6 prefix of an active lease entry that will
      be deleted from the relay.";
  }
}
output {
  leaf return-message {
    type string;
    description
      "Response message from the server. If available, a language
      identifier should be included in the message.";
    reference "BCP 14 (RFC 2277) IETF Policy on Character Sets
    and Languages, Section 4.2.";
  }
}
}

rpc clear-client-prefixes {
  nacm:default-deny-all;
  if-feature prefix-delegation;
  description
    "Clears all active prefix entries for a single client.";
  reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
  Section 4.4";
  input {
    leaf client-duid {
      type dhc6:duid;
      mandatory true;
      description
        "DUID of the client.";
    }
  }
  output {
    leaf return-message {
      type string;
      description
        "Response message from the server. If available, a
```

```
        language identifier should be included in the message.";
        reference "BCP 14 (RFC 2277) IETF Policy on Character Sets
        and Languages, Section 4.2.";
    }
}

rpc clear-interface-prefixes {
    nacm:default-deny-all;
    if-feature prefix-delegation;
    description
        "Clears all delegated prefix bindings from an
        interface on the relay.";
    reference "RFC8987: DHCPv6 Prefix Delegating Relay Requirements,
    Section 4.4";
    input {
        leaf interface {
            type leafref {
                path "/dhcpv6-relay/relay-if/if-name";
            }
            mandatory true;
            description
                "Reference to the relay interface that will have all
                active prefix delegation bindings deleted.";
        }
    }
    output {
        leaf return-message {
            type string;
            description
                "Response message from the server. If available, a
                language identifier should be included in the message.";
            reference "BCP 14 (RFC 2277) IETF Policy on Character Sets
            and Languages, Section 4.2.";
        }
    }
}

/*
 * Notifications
 */

notification relay-event {
    description
        "DHCPv6 relay event notifications.";
    container topology-change {
        description
            "Raised if the entry for an interface with DHCPv6 related
```

```

        configuration or state is removed from if:interface-refs.";
    leaf relay-if-name {
        type leafref {
            path "/dhcpv6-relay/relay-if/if-name";
        }
        description
            "Name of the interface that has been removed.";
    }
    leaf last-ipv6-addr {
        type inet:ipv6-address;
        description
            "Last IPv6 address configured on the interface.";
    }
}
}
}
<CODE ENDS>

```

#### 4.4. DHCPv6 Client YANG Module

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

```

<CODE BEGINS> file "ietf-dhcpv6-client@2022-03-07.yang"

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

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

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

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

    import ietf-interfaces {

```

```
    prefix if;
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }
```

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

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

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

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

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

```
revision 2022-03-07 {
  description
    "Initial Revision.";
  reference
    "XXXX: YANG Data Model for DHCPv6 Configuration";
```



```
}

/*
 * Features
 */

feature non-temp-addr {
  description
    "Denotes that the client supports DHCPv6 non-temporary address
    allocations.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 6.2";
}

feature temp-addr {
  description
    "Denotes that the client supports DHCPv6 temporary address
    allocations.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 6.5";
}

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

feature anon-profile {
  description
    "Denotes that the client supports DHCP anonymity profiles.";
  reference "RFC 7844: Anonymity Profiles for DHCP Clients";
}

/*
 * Groupings
 */

grouping message-statistics {
  description
    "Counters for DHCPv6 messages.";
  leaf discontinuity-time {
    type yang:date-and-time;
    description
      "The time on the most recent occasion at which any one or
      more of DHCPv6 client's counters suffered a discontinuity."
  }
}
```

```
        If no such discontinuities have occurred since the last
        re-initialization of the local management subsystem, then
        this node contains the time the local management subsystem
        re-initialized itself.";
    }
    leaf solicit-count {
        type yang:counter32;
        config "false";
        description
            "Number of Solicit (1) messages sent.";
    }
    leaf advertise-count {
        type yang:counter32;
        config "false";
        description
            "Number of Advertise (2) messages received.";
    }
    leaf request-count {
        type yang:counter32;
        config "false";
        description
            "Number of Request (3) messages sent.";
    }
    leaf confirm-count {
        type yang:counter32;
        config "false";
        description
            "Number of Confirm (4) messages sent.";
    }
    leaf renew-count {
        type yang:counter32;
        config "false";
        description
            "Number of Renew (5) messages sent.";
    }
    leaf rebind-count {
        type yang:counter32;
        config "false";
        description
            "Number of Rebind (6) messages sent.";
    }
    leaf reply-count {
        type yang:counter32;
        config "false";
        description
            "Number of Reply (7) messages received.";
    }
    leaf release-count {
```

```
    type yang:counter32;
    config "false";
    description
        "Number of Release (8) messages sent.";
}
leaf decline-count {
    type yang:counter32;
    config "false";
    description
        "Number of Decline (9) messages sent.";
}
leaf reconfigure-count {
    type yang:counter32;
    config "false";
    description
        "Number of Reconfigure (10) messages received.";
}
leaf information-request-count {
    type yang:counter32;
    config "false";
    description
        "Number of Information-request (11) messages sent.";
}
leaf discarded-message-count {
    type yang:counter32;
    config "false";
    description
        "Number of messages that have been discarded for any
        reason.";
}
}

grouping lease-state {
    description
        "Information about the active IA_NA lease.";
    leaf preferred-lifetime {
        type dhc6:timer-seconds32;
        description
            "The preferred lifetime for the leased address
            expressed in seconds.";
    }
    leaf valid-lifetime {
        type dhc6:timer-seconds32;
        description
            "The valid lifetime for the leased address expressed
            in seconds.";
    }
    leaf allocation-time {
```

```
    type yang:date-and-time;
    description
      "Time and date that the address was first leased.";
  }
  leaf last-renew-rebind {
    type yang:date-and-time;
    description
      "Time of the last successful renew or rebind of the
      leased address.";
  }
  leaf server-duid {
    type dhc6:duid;
    description
      "DUID of the leasing server.";
  }
  uses dhc6:status;
}

grouping option-request-option-group {
  description
    "OPTION_ORO (6) Option Request Option. A client MUST include
    an Option Request option in a Solicit, Request, Renew,
    Rebind, or Information-request message to inform the server
    about options the client wants the server to send to the
    client.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
  IPv6 (DHCPv6), Sections 21.23, 21.24, 21.25, & 21.7";
  container option-request-option {
    description
      "OPTION_ORO (6) Option Request Option.";
    leaf-list oro-option {
      type uint16;
      description
        "List of options that the client is requesting,
        identified by option code. This list MUST include the
        code for option SOL_MAX_RT (82) when included in a
        Solicit-message. If this option is being sent in an
        Information-request message, then the code for option
        OPTION_INFORMATION_REFRESH_TIME (32) and INF_MAX_RT (83)
        MUST be included.";
    }
  }
}

grouping user-class-option-group {
  description
    "OPTION_USER_CLASS (15) User Class Option";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
```

```
    IPv6 (DHCPv6), Section 21.15";
  container user-class-option {
    presence "Configures the option";
    description
      "OPTION_USER_CLASS (15) User Class Option.";
    list user-class-data-instance {
      key user-class-data-id;
      min-elements 1;
      description
        "The user classes of which the client is a member.";
      leaf user-class-data-id {
        type uint8;
        description
          "User class data ID";
      }
      leaf user-class-data {
        type binary;
        description
          "Opaque field representing a User Class of which the
            client is a member.";
      }
    }
  }
}

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

```
        "The vendor classes of which the client is a member.";
    leaf vendor-class-data-id {
        type uint8;
        description
            "Vendor class data ID";
    }
    leaf vendor-class-data {
        type binary;
        description
            "Opaque field representing a vendor class of which
            the client is a member.";
    }
}
}
}

/*
 * Data Nodes
 */

container dhcpv6-client {
    description
        "DHCPv6 client configuration and state.";
    leaf enabled {
        type boolean;
        default true;
        description
            "Globally enables the DHCP client function.";
    }
    leaf client-duid {
        if-feature "(non-temp-addr or prefix-delegation " +
            "or temp-addr) and not anon-profile";
        type dhc6:duid;
        description
            "A single Client DUID that will be used by all of the
            client's DHCPv6 enabled interfaces.";
        reference "RFC 8415: Dynamic Host Configuration Protocol for
            IPv6 (DHCPv6), Section 11";
    }
    list client-if {
        key if-name;
        description
            "The list of interfaces for which the client will
            be requesting DHCPv6 configuration.";
        leaf if-name {
            type if:interface-ref;
            mandatory true;
        }
    }
}
```

```
    description
      "Reference to the interface entry that the requested
       configuration is relevant to.";
  }
  leaf enabled {
    type boolean;
    default true;
    description
      "Enables the DHCP client function for this interface.";
  }
  leaf interface-duid {
    if-feature "(non-temp-addr or prefix-delegation " +
      "or temp-addr) and anon-profile";
    type dhc6:duid;
    description
      "Per-interface Client DUIDs for use with DHCP anonymity
       profiles.";
    reference "RFC 7844: Anonymity Profiles for DHCP Clients,
      Section 3";
  }
  container client-configured-options {
    description
      "Definitions for DHCPv6 options that can be be sent by
       the client. Additional option definitions can be
       augmented to this location from other YANG modules as
       required.";
    uses option-request-option-group;
    uses dhc6:rapid-commit-option-group;
    uses user-class-option-group;
    uses vendor-class-option-group;
    uses dhc6:vendor-specific-information-option-group;
    uses dhc6:reconfigure-accept-option-group;
  }
  list ia-na {
    if-feature non-temp-addr;
    key ia-id;
    description
      "Configuration relevant for an IA_NA (Identity Association
       for Non-temporary Addresses).";
    reference "RFC 8415: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6), Section 13.1";
    leaf ia-id {
      type uint32;
      description
        "A unique identifier for this IA_NA.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
        for IPv6 (DHCPv6), Section 12";
    }
  }
}
```

```
    container ia-na-options {
      description
        "An augmentation point for additional options
        that the client may send in the IA_NA-options field
        of OPTION_IA_NA.";
    }
    container lease-state {
      config false;
      description
        "Information about the active IA_NA lease.";
      leaf ia-na-address {
        type inet:ipv6-address;
        description
          "Address that is currently leased.";
      }
      leaf lease-t1 {
        type dhc6:timer-seconds32;
        description
          "The time interval after which the client should
          contact the server from which the addresses in the
          IA_NA were obtained to extend the lifetimes of the
          addresses assigned to the IA_NA.";
      }
      leaf lease-t2 {
        type dhc6:timer-seconds32;
        description
          "The time interval after which the client should
          contact any available server to extend the lifetimes
          of the addresses assigned to the IA_NA.";
      }
      uses lease-state;
    }
  }
  list ia-ta {
    if-feature temp-addr;
    key ia-id;
    description
      "Configuration relevant for an IA_TA (Identity Association
      for Temporary Addresses).";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 13.2";
    leaf ia-id {
      type uint32;
      description
        "The unique identifier for this IA_TA.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6), Section 12";
    }
  }
}
```



```
    container ia-ta-options {
      description
        "An augmentation point for additional options
        that the client may send in the IA_TA-options field
        of OPTION_IA_TA.";
    }
    container lease-state {
      config "false";
      description
        "Information about an active IA_TA lease.";
      leaf ia-ta-address {
        type inet:ipv6-address;
        description
          "Address that is currently leased.";
      }
      uses lease-state;
    }
  }
  list ia-pd {
    if-feature prefix-delegation;
    key ia-id;
    description
      "Configuration relevant for an IA_PD (Identity Association
      for Prefix Delegation).";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 13.3";
    leaf ia-id {
      type uint32;
      description
        "The unique identifier for this IA_PD.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6), Section 12";
    }
    leaf prefix-length-hint {
      type uint8 {
        range "1..128";
      }
      description
        "Prefix-length hint value included in the messages sent
        to the server to indicate a preference for the size of
        the prefix to be delegated.";
      reference "RFC 8415: Dynamic Host Configuration Protocol
      for IPv6 (DHCPv6), Section 18.2.1";
    }
  }
  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 lease-t1 {
            type dhc6:timer-seconds32;
            description
                "The time interval after which the client should
                contact the server from which the addresses in the
                IA_NA were obtained to extend the lifetimes of the
                addresses assigned to the IA_PD.";
        }
        leaf lease-t2 {
            type dhc6:timer-seconds32;
            description
                "The time interval after which the client should
                contact any available server to extend the lifetimes
                of the addresses assigned to the IA_PD.";
        }
        uses lease-state;
    }
}
container statistics {
    description
        "DHCPv6 message counters for the client.";
    uses message-statistics;
}
}

/*
 * Notifications
 */

notification invalid-ia-address-detected {
    if-feature "non-temp-addr or temp-addr";
    description
        "Notification sent when an address received in an identity
        association option is determined invalid. Possible conditions
        include a duplicate or otherwise illegal address.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
```

```
    IPv6 (DHCPv6), Section 18.2.10.1";
  leaf ia-id {
    type uint32;
    mandatory true;
    description
      "IA-ID";
  }
  leaf ia-na-t1-timer {
    type uint32;
    description
      "The value of the T1 time field for non-temporary address
      allocations (OPTION_IA_NA).";
  }
  leaf ia-na-t2-timer {
    type uint32;
    description
      "The value of the preferred-lifetime field for non-temporary
      address allocations (OPTION_IA_NA).";
  }
  leaf invalid-address {
    type inet:ipv6-address;
    description
      "The IP address which has been detected to be invalid.";
  }
  leaf preferred-lifetime {
    type uint32;
    description
      "The value of the preferred-lifetime field in
      OPTION_IAADDR.";
  }
  leaf valid-lifetime {
    type uint32;
    description
      "The value of the valid-lifetime field in OPTION_IAADDR.";
  }
  leaf ia-options {
    type binary;
    description
      "A copy of the contents of the IAaddr-options field.";
  }
  leaf description {
    type string;
    description
      "Description of the invalid Identity Association (IA)
      detection error.";
  }
}
```

```
notification transmission-failed {
  description
    "Notification sent when the transmission or retransmission
    of a message fails.";
  reference "RFC 8415: Dynamic Host Configuration Protocol for
  IPv6 (DHCPv6), Section 7.6";
  leaf failure-type {
    type enumeration {
      enum "solicit-timeout" {
        description
          "Max Solicit timeout value (SOL_MAX_RT) exceeded.";
      }
      enum "request-timeout" {
        description
          "Max Request timeout value (REQ_MAX_RT) exceeded.";
      }
      enum "request-retries-exceeded" {
        description
          "Max Request retry attempts (REC_MAX_RC) exceeded.";
      }
      enum "confirm-duration-exceeded" {
        description
          "Max Confirm duration (CNF_MAX_RD) exceeded.";
      }
      enum "renew-timeout" {
        description
          "Max Renew timeout value (REN_MAX_RT) exceeded.";
      }
      enum "rebind-timeout" {
        description
          "Max Rebind timeout value (REB_MAX_RT)
          exceeded.";
      }
      enum "info-request-timeout" {
        description
          "Max Information-request timeout value (INF_MAX_RT)
          exceeded.";
      }
      enum "release-retries-exceeded" {
        description
          "Max Release retry attempts (REL_MAX_RC) exceeded.";
      }
      enum "decline-retries-exceeded" {
        description
          "Max Decline retry attempts (DEC_MAX_RT) exceeded.";
      }
    }
  }
  mandatory true;
}
```

```
        description
            "Description of the failure.";
    }
    leaf description {
        type string;
        description
            "Information related to the failure, such as number of
            retries and timer values.";
    }
}

notification unsuccessful-status-code {
    description
        "Notification sent when the client receives a message that
        includes an unsuccessful Status Code option.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 21.13";
    leaf server-duid {
        type dhcp6:duid;
        mandatory true;
        description
            "DUID of the server sending the unsuccessful error code.";
    }
    uses dhcp6:status;
}

notification server-duid-changed {
    if-feature "non-temp-addr or prefix-delegation or " +
        "temp-addr";
    description
        "Notification sent when the client receives a lease from a
        server with different DUID to the one currently stored by the
        client, e.g., in response to a Rebind message.";
    reference "RFC 8415: Dynamic Host Configuration Protocol for
    IPv6 (DHCPv6), Section 18.2.5";
    leaf new-server-duid {
        type dhcp6:duid;
        mandatory true;
        description
            "DUID of the new server.";
    }
    leaf previous-server-duid {
        type dhcp6:duid;
        mandatory true;
        description
            "DUID of the previous server.";
    }
    leaf lease-ia-na {
```

```
    if-feature non-temp-addr;
    type leafref {
      path "/dhcpv6-client/client-if/ia-na/ia-id";
    }
    description
      "Reference to the IA_NA lease.";
  }
  leaf lease-ia-ta {
    if-feature temp-addr;
    type leafref {
      path "/dhcpv6-client/client-if/ia-ta/ia-id";
    }
    description
      "Reference to the IA_TA lease.";
  }
  leaf lease-ia-pd {
    if-feature prefix-delegation;
    type leafref {
      path "/dhcpv6-client/client-if/ia-pd/ia-id";
    }
    description
      "Reference to the IA_PD lease.";
  }
}
}
<CODE ENDS>
```

## 5. 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.

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

An attacker with read/write access to the DHCPv6 server can undertake various attacks, such as:

- \* Denial of service attacks, such as disabling the DHCP server service, or removing address/prefix pool configuration.
- \* Various attacks based on re-configuring the contents of DHCPv6 options, leading to several types of security or privacy threats. These options could redirect clients to services under an attacker's control. For example, changing the address of a DNS server supplied in a DHCP option to point to a rogue server.

An attacker sending DHCPv6 messages which cause the server to generate 'invalid-client-detected' and 'decline-received' notifications could be used as a DoS attack. Such an attack could be mitigated by the NETCONF client unsubscribing from the affected notifications.

An attacker with read/write access the DHCPv6 relay can undertake various attacks, such as:

- \* Denial of service attacks, based on disabling the DHCP relay function, or modifying the relay's "destination-address" to a non-existent address.
- \* Modifying 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., via get, get-config, or notification) to these data nodes. These subtrees and data nodes can be misused to track the activity or fingerprint the device type of the host:

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

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

Information about a server's configured address and prefix pools may be used by an attacker for network reconnaissance [RFC7707]. The following subtrees and data nodes could be used for this purpose:

- \* Information about client address allocation ranges: (dhc6-srv/  
allocation-ranges/allocation-range/address-pools/ address-pool/  
pool-prefix)
- \* Information about client prefix allocation ranges: (dhc6-srv/  
allocation-ranges/allocation-range/prefix-pools/ prefix-pool/pool-  
prefix)

[RFC7844] describes anonymity profiles for DHCP clients. These can be used to prevent client tracking on a visited network. Support for this can be enabled by implementing the 'anon-profile' feature in the client module.

[RFC7824] covers privacy considerations for DHCPv6 and is applicable here.

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

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

## 6. IANA Considerations

This document registers four URIs and four YANG modules.

### 6.1. URI Registration

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

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

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

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



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

## 6.2. YANG Module Name Registration

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

name: ietf-dhcpv6-server  
namespace: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-server  
prefix: dhc6-srv  
reference: RFC XXXX YANG Data Model for DHCPv6 Configuration

name: ietf-dhcpv6-relay  
namespace: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay  
prefix: dhc6-rly  
reference: RFC XXXX YANG Data Model for DHCPv6 Configuration

name: ietf-dhcpv6-client  
namespace: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client  
prefix: dhc6-clnt  
reference: RFC XXXX YANG Data Model for DHCPv6 Configuration

name: ietf-dhcpv6-common  
namespace: urn:ietf:params:xml:ns:yang:ietf-dhcpv6-common  
prefix: dhc6  
reference: RFC XXXX YANG Data Model for DHCPv6 Configuration

## 7. Acknowledgments

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

## 8. Contributors

The following individuals are co-authors of this document:

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

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

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

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

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

## 9. References

### 9.1. 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>>.
- [RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, DOI 10.17487/RFC2277, January 1998, <<https://www.rfc-editor.org/info/rfc2277>>.
- [RFC3118] Droms, R., Ed. and W. Arbaugh, Ed., "Authentication for DHCP Messages", RFC 3118, DOI 10.17487/RFC3118, June 2001, <<https://www.rfc-editor.org/info/rfc3118>>.

- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6355] Narten, T. and J. Johnson, "Definition of the UUID-Based DHCPv6 Unique Identifier (DUID-UUID)", RFC 6355, DOI 10.17487/RFC6355, August 2011, <<https://www.rfc-editor.org/info/rfc6355>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <<https://www.rfc-editor.org/info/rfc6242>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <<https://www.rfc-editor.org/info/rfc7844>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <<https://www.rfc-editor.org/info/rfc8341>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <<https://www.rfc-editor.org/info/rfc8343>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [RFC8987] Farrer, I., Kottapalli, N., Hunek, M., and R. Patterson, "DHCPv6 Prefix Delegating Relay Requirements", RFC 8987, DOI 10.17487/RFC8987, February 2021, <<https://www.rfc-editor.org/info/rfc8987>>.
- [IANA-HARDWARE-TYPES] Internet Assigned Numbers Authority, "Hardware Types", <<https://www.iana.org/assignments/arp-parameters>>.
- [IANA-PEN] Internet Assigned Numbers Authority, "Private Enterprise Numbers", <<https://www.iana.org/assignments/enterprise-numbers>>.
- [IANA-DHCPV6-OPTION-CODES] Internet Assigned Numbers Authority, "DHCPv6 Option Codes", <<https://www.iana.org/assignments/dhcpv6-parameters>>.
- [IANA-DHCP-AUTH-NAMESPACES] Internet Assigned Numbers Authority, "Dynamic Host Configuration Protocol (DHCP) Authentication Option Name Spaces", <<https://www.iana.org/assignments/auth-namespaces>>>.

## 9.2. 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>>.
- [RFC7707] Gont, F. and T. Chown, "Network Reconnaissance in IPv6 Networks", RFC 7707, DOI 10.17487/RFC7707, March 2016, <<https://www.rfc-editor.org/info/rfc7707>>.
- [RFC7824] Krishnan, S., Mrugalski, T., and S. Jiang, "Privacy Considerations for DHCPv6", RFC 7824, DOI 10.17487/RFC7824, May 2016, <<https://www.rfc-editor.org/info/rfc7824>>.
- [I-D.ietf-netconf-tls-client-server] Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-26, 14 December 2021, <<https://tools.ietf.org/html/draft-ietf-netconf-tls-client-server-26>>.

## Appendix A. Data Tree Examples

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

### A.1. DHCPv6 Server Configuration Examples

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

- \* Enabling the DHCP server function.
- \* The server's DUID.
- \* An option set (id=1) with configuration for the Solicit Max Retry Timeout (SOL\_MAX\_RT (82)) option.
- \* A single network range (2001:db8::/32).
- \* A single address pool, with start and end addresses, relevant lease timers and an option-set-id of "1" referencing the option set configured above.

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

Figure 4: Basic Server Configuration Example XML

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

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

Figure 5: Server Host Reservation Configuration Example XML Snippet

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

The 'max-pd-space-utilization' is set to 80 percent so that a 'prefix-pool-utilization-threshold-exceeded' notification will be raised if the number of prefix allocations exceeds this.

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

Figure 6: Server Prefix Delegation Configuration Example XML Snippet

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

The example shows how the option definitions can be extended via augmentation. In this case, "OPTION\_SIP\_SERVER\_D (21) SIP Servers Domain-Name List" from the example module in Appendix B has been augmented to the server's option set.



```

<option-sets>
  <option-set>
    <option-set-id>1</option-set-id>
    <description>Example DHCP option set</description>
    <vendor-specific-information-options>
      <vendor-specific-information-option>
        <enterprise-number>32473</enterprise-number>
        <vendor-option-data>
          <sub-option-code>01</sub-option-code>
          <sub-option-data>1234abcd</sub-option-data>
        </vendor-option-data>
        <vendor-option-data>
          <sub-option-code>02</sub-option-code>
          <sub-option-data>abcd1234</sub-option-data>
        </vendor-option-data>
      </vendor-specific-information-option>
    </vendor-specific-information-options>
    <sol-max-rt-option>
      <sol-max-rt-value>3600</sol-max-rt-value>
    </sol-max-rt-option>
    <sip-server-domain-name-list-option
      xmlns="https://example.com/ns/example-dhcpv6-opt-sip-serv">
      <sip-server>
        <sip-serv-id>0</sip-serv-id>
        <sip-serv-domain-name>sip1.example.org</sip-serv-domain-name>
      </sip-server>
      <sip-server>
        <sip-serv-id>1</sip-serv-id>
        <sip-serv-domain-name>sip2.example.org</sip-serv-domain-name>
      </sip-server>
    </sip-server-domain-name-list-option>
  </option-set>
</option-sets>

```

Figure 7: Server Option Set Configuration Example XML Snippet

#### A.2. DHCPv6 Relay Configuration Example

The following example shows a basic configuration for a single DHCP relay interface and its interaction with the ietf-interfaces module. The configuration shows two XML documents, one for ietf-interfaces and a second for ietf-dhcpv6-relay, defining:

- \* Configuring an interface using the ietf-interfaces module that the relay configuration will be applied to.
- \* Enabling the DHCP relay function globally and for the relevant interface.

- \* Referencing the interface that the relay configuration is relevant for via an interface-ref to the ietf-interfaces module.
- \* Defining two destination addresses that incoming DHCP messages will be relayed to.
- \* Configures the link-address value that will be sent in the relay-forward message.
- \* Configuring a value for the Interface ID Option (OPTION\_INTERFACE\_ID (18)), which will be included in the relay forward message.

```

<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <description>DHCPv6 Relay Interface</description>
    <enabled>true</enabled>
  </interface>
</interfaces>

<dhcpv6-relay xmlns="urn:ietf:params:xml:ns:yang:ietf-dhcpv6-relay">
  <enabled>true</enabled>
  <relay-if>
    <if-name>eth0</if-name>
    <enabled>true</enabled>
    <destination-address>2001:db8:2::1</destination-address>
    <destination-address>2001:db8:2::2</destination-address>
    <link-address>2001:db8:3::1</link-address>
    <relay-options>
      <interface-id-option>
        <interface-id>EXAMPLEINTERFACEID01</interface-id>
      </interface-id-option>
    </relay-options>
  </relay-if>
</dhcpv6-relay>

```

Figure 8: Basic Relay Configuration Example XML

### A.3. DHCPv6 Client Configuration Example

The following example shows a basic configuration for a DHCP client and its interaction with the ietf-interfaces module. The configuration shows two XML documents, one for ietf-interfaces and a second for ietf-dhcpv6-client defining:

- \* Configuring an interface using the ietf-interfaces module that the client configuration will be applied to.
- \* Enabling the DHCP client function globally and for the relevant interface.
- \* References the interface that the client configuration is relevant for via an interface-ref to the ietf-interfaces module.
- \* Sets the DUID for the DHCPv6 enabled interface.
- \* Configures a list of option codes that will be requested by the client in its Option Request Option (OPTION\_ORO (5)).
- \* Configures a single instance of the Vendor-specific Information Option (OPTION\_VENDOR\_OPTS (17)) with a single sub-option data item.
- \* Requests a non-temporary IPv6 address (IA\_NA) with an identity association interface identifier of 1.
- \* Requests an IPv6 delegated prefix address (IA\_PD) with an identity association interface identifier of 2.

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces"
  xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">
  <interface>
    <name>eth0</name>
    <type>ianaift:ethernetCsmacd</type>
    <description>DHCPv6 Relay Interface</description>
    <enabled>true</enabled>
  </interface>
</interfaces>

<dhcpv6-client
  xmlns="urn:ietf:params:xml:ns:yang:ietf-dhcpv6-client">
  <enabled>true</enabled>
  <client-if>
    <if-name>eth0</if-name>
    <enabled>true</enabled>
    <interface-duid>000200090CC084D303000913</interface-duid>
    <client-configured-options>
      <option-request-option>
        <oro-option>17</oro-option>
        <oro-option>23</oro-option>
        <oro-option>24</oro-option>
        <oro-option>82</oro-option>
      </option-request-option>
      <vendor-specific-information-options>
        <vendor-specific-information-option>
          <enterprise-number>32473</enterprise-number>
          <vendor-option-data>
            <sub-option-code>1</sub-option-code>
            <sub-option-data>abcd1234</sub-option-data>
          </vendor-option-data>
        </vendor-specific-information-option>
      </vendor-specific-information-options>
    </client-configured-options>
    <ia-na>
      <ia-id>1</ia-id>
    </ia-na>
    <ia-pd>
      <ia-id>2</ia-id>
    </ia-pd>
  </client-if>
</dhcpv6-client>
```

Figure 9: Basic Client Configuration Example XML

## Appendix B. Example of Augmenting Additional DHCPv6 Option Definitions

The following section provides an 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]. Example XML configuration, showing the interworking with other modules is provided in Figure 7.

The module is constructed as follows:

- \* The module is named using a meaningful, shortened version of the document name in which the DHCP option format is specified.
- \* A separate grouping is used to define each option.
- \* The name of the option is taken from the registered IANA name for the option, with an '-option' suffix added.
- \* The description field is taken from the relevant option code name and number.
- \* The reference section is the number and name of the RFC in which the DHCPv6 option is defined.
- \* The remaining fields match the fields in the DHCP option. They are in the same order as defined in the DHCP option. Where-ever possible, the format that is defined for the DHCP field should be matched by the relevant YANG type.
- \* Fields which can have multiple entries or instances are defined using list or leaf-list nodes.

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

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

```
}

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

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

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

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

  Copyright (c) 2022 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 Revised BSD License set
  forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents
  (https://trustee.ietf.org/license-info).https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
  for full legal notices.";

revision 2022-03-07 {
  description
    "Initial Revision.";
  reference
    "XXXX: YANG Data Model for DHCPv6 Configuration";
}

/*
 * Groupings
 */
```

```
grouping sip-server-domain-name-list-option-group {
  description
    "OPTION_SIP_SERVER_D (21) SIP Servers Domain-Name List";
  reference "RFC 3319: Dynamic Host Configuration Protocol
    (DHCPv6) Options for Session Initiation Protocol (SIP)
    Servers";
  container sip-server-domain-name-list-option {
    description
      "OPTION_SIP_SERVER_D (21) SIP Servers Domain Name List
      Option.";
    list sip-server {
      key sip-serv-id;
      description
        "SIP server information.";
      leaf sip-serv-id {
        type uint8;
        description
          "SIP server list identifier.";
      }
      leaf sip-serv-domain-name {
        type inet:domain-name;
        description
          "SIP server domain name.";
      }
    }
  }
}

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

```

        type inet:ipv6-address;
        description
            "SIP server IPv6 address.";
    }
}
}
}

/*
 * Augmentations
 */

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

```

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

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

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

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

```
'/dhc6-srv:dhc6-srv/dhc6-srv:allocation-ranges/dhc6-srv:allocation-
range/dhc6-srv:prefix-pools/dhc6-srv:prefix-pool'
```

#### Appendix C. Example Vendor Specific Server Configuration Module

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



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

For simplicity, this example module assumes that the DHCPv6 server is colocated with the MySQL or PostgreSQL database server and can serve traffic securely on the localhost without additional cryptographic protection. In a production deployment, these functions would likely not be colocated and thus use TLS to secure the database connection between the DHCPv6 server and database server. A YANG module for configuring TLS is defined in [I-D.ietf-netconf-tls-client-server].

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

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

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

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

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

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

## description

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

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

## revision 2022-03-07 {

## description

"Initial Revision.";

## reference

"XXXX: YANG Data Model for DHCPv6 Configuration";

}

/\*

\* Groupings

\*/

## grouping config {

## description

"Parameters necessary for the configuration of a DHCPv6 server";

## container serv-attributes {

## description

"Contains basic attributes necessary for running a DHCPv6 server.";

## leaf name {

type string;

## description

"Name of the DHCPv6 server.";

}

## leaf description {

type string;

## description

```
        "Description of the DHCPv6 server.";
    }
    leaf ipv6-listen-port {
        type uint16;
        default 547;
        description
            "UDP port that the server will listen on.";
    }
    choice listening-interfaces {
        default all-interfaces;
        description
            "Configures which interface or addresses the server will
            listen for incoming messages on.";
        case all-interfaces {
            container all-interfaces {
                presence true;
                description
                    "Configures the server to listen for incoming messages
                    on all IPv6 addresses (unicast and multicast) on all of
                    its network interfaces.";
            }
        }
        case interface-list {
            leaf-list interfaces {
                type if:interface-ref;
                description
                    "List of interfaces on which the server will listen
                    for incoming messages. Messages addressed to any
                    valid IPv6 address (unicast and multicast) will be
                    received.";
            }
        }
        case address-list {
            leaf-list address-list {
                type inet:ipv6-address;
                description
                    "List of IPv6 address(es) on which the server will
                    listen for incoming DHCPv6 messages.";
            }
        }
    }
    leaf-list interfaces-config {
        type if:interface-ref;
        default "if:interfaces/if:interface/if:name";
        description
            "A leaf list of interfaces on which the server should
            listen.";
    }
}
```

```
container lease-storage {
  description
    "Configures how the server will store leases.";
  choice storage-type {
    description
      "The type of storage that will be used for lease
      information.";
    case memfile {
      description
        "Configuration for storing leases information in a
        Comma-Separated Value (CSV) file.";
      leaf memfile-name {
        type string;
        description
          "Specifies the absolute location of the lease file.
          The format of the string follow the semantics of
          the relevant operating system.";
      }
      leaf memfile-lfc-interval {
        type uint64;
        description
          "Specifies the interval in seconds, at which the
          server will perform a lease file cleanup (LFC).";
      }
    }
    case mysql {
      leaf mysql-name {
        type string;
        description
          "Name of the MySQL database, running on the
          localhost.";
      }
      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 3306;
        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 {
    leaf postgresql-name {
        type string;
        description
            "Name of the PostgreSQL database, running on the
            localhost.";
    }
    leaf postgresql-username {
        type string;
        description
            "User name of the account under which the server
            will access the database";
    }
    leaf postgresql-password {
        type string;
        description
            "Password of the account under which the server
            will access the database";
    }
    leaf postgresql-port {
        type inet:port-number;
        default 5432;
        description
            "If the database is located on a different system,
            the port number may be specified";
    }
    leaf postgresql-lfc-interval {
        type uint64;
        description
            "Specifies the interval in seconds, at which the
            server will perform a lease file cleanup (LFC)";
    }
}
```

```

    }
    leaf postgresql-connect-timeout {
        type uint64;
        description
            "Defines the timeout interval for connecting to the
            database. A longer interval can be specified if the
            database is remote.";
    }
}
}
}
}
}

/*
 * Augmentations
 */

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

```

## Appendix D. Example definition of class-selector configuration

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

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

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

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

- \* client-class leafrefs augmented to "allocation-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 a given allocation-range/pool if the class name matches.

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

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-dhcpv6-common {
    prefix dhc6;
  }

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

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

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

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

only.

Copyright (c) 2022 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 Revised BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.";

```
revision 2022-03-07 {
  description
    "Initial Revision.";
  reference
    "XXXX: YANG Data Model for DHCPv6 Configuration";
}

/*
 * Groupings
 */

grouping client-class-id {
  description
    "Definitions of client message classification for
    authorization and assignment purposes.";
  leaf client-class-name {
    type string;
    mandatory true;
    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 a received instance of
      OPTION_INTERFACE_ID (18).";
    leaf relay-interface {
      type string;
      description
        "An opaque value of arbitrary length generated by the
        relay agent to identify one of the relay agent's
        interfaces.";
    }
  }
  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
        "User Class value to match.";
    }
  }
  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 {
  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 enterprise-number {
      type uint32;
      description
        "The vendor's registered Enterprise Number as
        maintained by IANA.";
    }
    leaf vendor-class-data-id {
      type uint8;
      description
        "Vendor class data ID";
    }
    leaf vendor-class-data {
      type string;
      description
        "Opaque field for matching the client's vendor class
        data.";
    }
  }
}
case client-duid-id {
  description
    "Client class selection based on the value of the
    received client DUID.";
  leaf duid {
    type dhcp6:duid;
    description
      "Client DUID.";
  }
}
```

```
    }
  }
}

/*
 * Augmentations
 */

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

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

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

```
        path "/dhc6-srv:dhcpv6-server/dhc6-srv:" +
            "class-selector/client-classes/class/client-class-name";
    }
    description
        "Leafrefs to client classes.";
}

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

#### Author's Address

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

Email: [ian.farrer@telekom.de](mailto:ian.farrer@telekom.de)

Dynamic Host Configuration (DHC)  
Internet-Draft  
Intended status: Informational  
Expires: 23 May 2022

G.R. Ren  
L.H. He  
Y.L. Liu  
Tsinghua University  
19 November 2021

DHCPv6 Extension Practices and Considerations  
draft-ietf-dhc-problem-statement-of-mredhcpv6-08

Abstract

IP addresses assume an increasing number of attributes as communication identifiers to meet different requirements. Privacy protection, accountability, security, and manageability of networks can be supported by extending the DHCPv6 protocol as required. This document provides current extension practices and typical DHCPv6 server software in terms of extensions, defines a general model of DHCPv6, discusses some extension points, and presents extension cases.

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 23 May 2022.

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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	4
3. Current Extension Practices . . . . .	4
3.1. Standardized and Non-standardized DHCPv6 Extension Cases . . . . .	4
3.2. Current DHCPv6 Server Software Cases . . . . .	4
4. Extension Discussion . . . . .	5
4.1. DHCPv6 General Model . . . . .	5
4.2. Extension Points . . . . .	6
4.2.1. Messages . . . . .	6
4.2.2. Options . . . . .	6
4.2.3. Message Processing Functions . . . . .	7
4.2.4. Address Generation Mechanisms . . . . .	7
4.3. Extension Principles . . . . .	8
5. Extension Cases . . . . .	8
5.1. Software Configurations . . . . .	9
5.2. Option Definition and Server Modification . . . . .	9
5.3. Message Definition . . . . .	9
6. Security Considerations . . . . .	10
7. IANA Considerations . . . . .	10
8. Acknowledgements . . . . .	10
9. References . . . . .	10
9.1. Normative References . . . . .	10
9.2. Informative References . . . . .	10
Authors' Addresses . . . . .	14

## 1. Introduction

IP addresses play an essential role in communication over the Internet. Their generation and assignment are also closely linked to the privacy protection, accountability, security, and manageability of the network [I-D.gont-v6ops-ipv6-addressing-considerations]. The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC8415] is an important network protocol that can be used to dynamically provide IPv6 addresses and other network configuration parameters to IPv6 nodes. DHCPv6 can be continuously extended and improved through new options, protocols, and message processing mechanisms.

IP addresses assume an increasing number of properties as communication identifiers to meet different requirements. For example, APNA [APNA] and PAVI [PAVI] use addresses to enhance source responsibility and privacy protection. These requirements often need

to be reflected by IP address assignment protocols such as DHCPv6. Therefore, extensions to DHCPv6 are made to meet a wide variety of requirements, which is referred to as multi-requirement extensions to DHCPv6. However, it is not easy to extend DHCPv6 to meet a variety of requirements. Although DHCPv6 offers increasingly comprehensive functionality and DHCPv6 server software provides extension interfaces that allow administrators to change and customize the way they process and respond to DHCPv6 messages, there is still a lack of comprehensive understanding of where and how to extend in DHCPv6 effectively. Therefore, a detailed analysis is needed to clarify the issues and design principles and extract and unify design specifications to help better address the multi-demand scaling problem.

In summary, with the large-scale deployment and application of IPv6, new scenarios such as Data Center Network, Internet of Things, Industrial Internet, and Integrated satellite-terrestrial networks put forward new requirements for IP address allocation, e.g., the scale of address allocation, the efficiency of address update and synchronization, the address generation algorithms (such as association with location, identifier, and other information), and the scope of dynamic address configuration service relay and collaboration. At the same time, it also puts forward new requirements in network security, accountability, manageability, and privacy protection. These are what we call "multiple requirements". Multi-requirement extensions for DHCPv6 is to meet new scenarios and new requirements through the expansion of new messages, options, message processing functions, or address generation mechanisms for DHCPv6. Based on careful design principles, interfaces can be defined to support more customized multi-requirement extensions without sacrificing the stability of DHCPv6.

Some people would suggest that administrators modify the open-source DHCPv6 server to solve their problems. However, it takes considerable time to understand the code of an open-source DHCPv6 server, not to mention the time-consuming task of debugging errors, failures, or system crashes caused by modifying complex modules. Another problem is that as open-source software evolves, the source code of the server software may change (new features or bug fixes). Once the latest version of the open-source server software comes out [kea\_dhcp\_hook\_developers\_guide], users may need to rewrite their code. Therefore, the multi-requirement extensions to DHCPv6 to address the specific issues of administrators are essential and significant.



This document provides a survey of current extension practices and typical DHCPv6 server softwares on extensions and gives DHCPv6 extension considerations by defining a DHCPv6 general model, discussing the extension problems, and presenting extension cases.

## 2. Terminology

Familiarity with DHCPv6 and its terminology, as defined in [RFC8415], is assumed.

**Multi-requirement extensions:** The multi-requirement extensions for DHCPv6 is to meet new scenarios and requirements by extending DHCPv6 with new messages, options, message processing features, or address generation mechanisms.

## 3. Current Extension Practices

### 3.1. Standardized and Non-standardized DHCPv6 Extension Cases

Many documents attempt to extend DHCPv6. They can be classified into three categories.

Extended options	Most extensions for DHCPv6 are implemented in this way. New-defined options carry specific parameters in DHCPv6 messages, which helps DHCPv6 clients or servers know the detailed situation with each other.
Extended messages	Some documents define new protocols that aim to achieve specific goals, e.g., active leasequery [RFC7653], General Address Generation and Management System [GAGMS].
Extended entities	Some documents introduce third-party entities into the communications of DHCPv6 to achieve specific goals and provide better services, e.g., authentication [RFC7037].

### 3.2. Current DHCPv6 Server Software Cases

A lot of commercial and open source DHCPv6 servers exist, including Cisco Prime Network Registrar (CPNR) DHCP [CPNR], DHCP Broadband [DHCP\_Broadband], FreeRADIUS DHCP [FreeRADIUS\_DHCP], ISC DHCP [ISC\_DHCP], Kea DHCP [Kea\_DHCP], Microsoft DHCP [Microsoft\_DHCP], Nominum DHCP [Nominum\_DHCP], VitalQIP [VitalQIP], and WIDE DHCPv6 [WIDE\_DHCPv6]. Commercial and open-source DHCPv6 software often considers the extensions of DHCPv6 servers because they cannot always meet the requirements that the administrators want. For example,

CPNR DHCP server provides extension APIs and allows administrators to write extensions and functions to alter and customize how it handles and responds to DHCP requests. A network operator usually decides what packet process to modify, how to modify, and which extension point to attach the extension. Then the network operator writes the extension and adds the well-written extension to the extension point of the DHCP server. Finally, the network operator reloads the DHCP server and debugs whether the server runs as it expects. Similarly, Kea DHCP provides hook mechanisms, a well-designed interface for third-party code, to solve the problem that the DHCP server does not quite do what a network operator require.

#### 4. Extension Discussion

This section elaborates multi-requirement extensions for DHCPv6. Section 4.1 describes the general model of DHCPv6, while Section 4.2 analyzes the extension points and requirements.

##### 4.1. DHCPv6 General Model

Figure 1 summarizes the DHCPv6 general model and its possible extensions: messages, options, message processing functions, and address generation mechanisms.

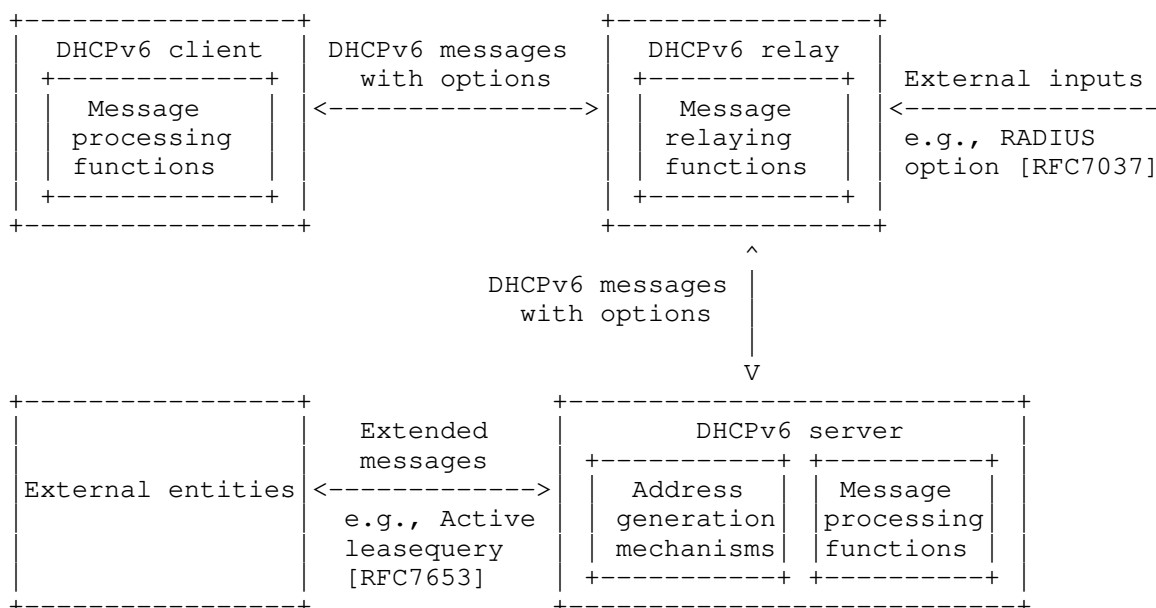


Figure 1: DHCPv6 general model and its possible extensions.

## 4.2. Extension Points

### 4.2.1. Messages

On the one hand, new messages can be designed and added to the DHCPv6 protocol to enrich its functionalities. For example, [RFC5007] defines new leasequery messages to allow a requestor to retrieve information on the bindings for a client from one or more servers. [RFC5460] expands on the Leasequery protocol by defines new messages and allowing for bulk transfer of DHCPv6 binding data via TCP. [RFC7653] defines active leasequery messages to keep the requestor up to date with DHCPv6 bindings. [RFC8156] defines failover messages to provide a mechanism for running two servers with the capability for either server to take over clients' leases in case of server failure or network partition.

On the other hand, people are concerned about the security and privacy issues of the DHCPv6 protocol. [RFC7824] describes the privacy issues associated with the use of DHCPv6, respectively. DHCPv6 does not provide privacy protection on messages and options. Other nodes can see the options transmitted in DHCPv6 messages between DHCPv6 clients and servers. Extended messages can be designed to secure exchanges between DHCPv6 entities.

### 4.2.2. Options

DHCPv6 allows defining options to transmit parameters between DHCPv6 entities for common requirements, e.g., DNS configurations [RFC3646], NIS configurations [RFC3898], SNTP configurations [RFC4075], relay agent subscriber-id [RFC4580], relay agent remote-id [RFC4649], FQDN configurations [RFC4704], relay agent echo request [RFC4994], network boot [RFC5970], Relay-Supplied Options [RFC6422], virtual subnet selection [RFC6607], client link-layer address [RFC6939], and software source binding prefix hint [RFC8539]. Also, these parameters may come from external entities. For example, [RFC7037] defines RADIUS option to exchange authorization and identification information between the DHCPv6 relay agent and DHCPv6 server.

In other cases, network operators may require DHCPv6 messages to transmit some self-defined options between clients and servers. Currently, the vendor-specific information option allows clients and servers to exchange vendor-specific information. Therefore, administrative domains can define and use the sub-options of the vendor-specific information option to serve their private purposes. The content of the self-defined options may come from two sources: devices and users. If the content of self-defined options comes from users, two methods can be used to solve the problem. The first one is that the clients provide related interfaces to receive such

information, which is currently merely supported. The second one is that DHCPv6 relays obtain such information and add it to the clients' requests. But this always depends on other protocols to allow DHCPv6 relays to get the information first.

#### 4.2.3. Message Processing Functions

Although current commercial or open-source DHCPv6 server softwares provide comprehensive functionalities, they still cannot meet all customers' requirements of processing DHCPv6 requests. Therefore, they will offer interfaces that customers can use to write their specific extensions to affect the way how DHCPv6 servers handle and respond to DHCP requests. For example, a network operator may want his DHCPv6 server to communicate with external servers. Thus, he may alter his DHCPv6 server through the given extensions to achieve such a goal. However, not all DHCPv6 software considers this extension.

#### 4.2.4. Address Generation Mechanisms

Currently, the DHCPv6 servers assign addresses, prefixes and other configuration options according to their configured policies. Generally, different networks may prefer different address generation mechanisms. Several address generation mechanisms for SLAAC [RFC4862] (e.g., IEEE 64-bit EUI-64 [RFC2464], Constant, semantically opaque [Microsoft], Temporary [RFC4941], and Stable, semantically opaque [RFC7217]) proposed for different requirements can be utilized in DHCPv6 protocol as well. Note that [RFC7943] is the DHCPv6 version of Stable, semantically opaque [RFC7217]. The many types of IPv6 address generation mechanisms available have brought about flexibility and diversity. Therefore, corresponding interfaces could be open and defined to allow other address generation mechanisms to be configured.

Moreover, several basic operations are defined to support the design of IPv6 addresses generation mechanisms. A new IPv6 address generation mechanism can be made up of the combination of the following basic operations. Also, new basic operations can be defined to support new functions.

Invert(x, n)	invert bit n of input x.
Insert(x, n, s)	insert s after bit n of input x.
Concatenate(x, y, ...)	concatenate input [x, y, ...] sequentially.
Replace(x, n, m, s)	change from bit n to bit m of input x into s.

Note that the length of  $s$  must be equal to  $m-n+1$ . When  $n=m$ , change only one bit of input  $x$ .

`Truncate(x, n, m)`      truncate from bit  $n$  to bit  $m$  of input  $x$  as the output

`Encrypt(x, k)`          use some specific encryption algorithm to encrypt input  $x$  with key  $k$ . Encryption algorithms can be IDEA, AES, RSA, etc.

`Hash(x)`                calculate the hash digest value of input  $x$ . Hash algorithms can be MD5, SHA1, SHA256, etc.

For example, temporary addresses in [RFC4941] can be expressed as `tempAddr(eui64, history) = Replace(Truncate(Hash(Concatenate(eui64, history)), 0, 63), 6, 6, 0)`, where `eui64` means the EUI-64 identifier defined in [RFC2464] and `history` means a history value defined in [RFC4941].

#### 4.3. Extension Principles

The principles used to conduct multi-requirement extensions for DHCPv6 are summarized as follows:

- 1) Do not change the basic design of DHCPv6.
- 2) Use simpler interfaces to define and support more extensions.

#### 5. Extension Cases

Administrative domains may enforce local policies according to their requirements, e.g., authentication, accountability. Several kinds of multi-requirement extensions are presented in this section, including configurations in current DHCPv6 software, option definition and server modification, and message definition between DHCPv6 entities and third-party entities. IPv6 addresses are related to manageability, security, traceability, and accountability of networks. As DHCPv6 assigns IPv6 addresses to IPv6 nodes, it is important that DHCPv6 provides interfaces to allow administrative domains to conduct extensions to meet their multi-requirements.

### 5.1. Software Configurations

Currently, many DHCPv6 servers provide administrative mechanisms, e.g., host reservation and client classification. Host reservation is often used to assign certain parameters (e.g., IP addresses) to specific devices. For example, a client with special access rights (e.g., a firewall rule that allows access based on the source's IP address) needs to keep its address allowed in the firewall configuration. Another use case is a device with a mission-critical network service that needs access by IP address in case a DNS lookup fails. Client classification is often used to differentiate between different types of clients and treat them accordingly in certain cases. This classification allows DHCP addresses or options to be assigned based on specific device characteristics or some network identifier. Grouping devices by client class makes it more convenient to perform bulk configuration settings. A typical example is the network access security policy. For example, a client class can be configured so that devices in that class are assigned IP addresses in subnets that are restricted to the public Internet due to security policies applied to the subnet/network on the router or firewall.

### 5.2. Option Definition and Server Modification

More complicated extensions of DHCPv6 are needed to meet specific requirements. For example, considering such a requirement that DHCPv6 servers assign IPv6 addresses generated by user identifiers to the clients in a network to hold users accountable, two extensions should be fulfilled to meet this requirement. The first one is that clients send their user identifiers to servers. This can be achieved by defining and using sub-options of vendor-specific information option. The second one is that servers use user identifiers to generate IP addresses. To achieve this goal, extension mechanisms provided by the server software such as extension points in CPNR [CPNR] and hook mechanisms in Kea DHCP [Kea\_DHCP] can be used.

### 5.3. Message Definition

Some extensions for DHCPv6 may need the support of third-party entities. For example, [RFC7037] introduces RADIUS entities into the message exchanges between DHCPv6 entities for better service provision. The authentication in [RFC7037] can also be used to meet the accountability requirement mentioned above because it is important to authenticate users first before assigning IP addresses generated from user identifiers. Usually, this kind of extension requires the definition of messages communicated between DHCPv6 entities and third-party entities, e.g., active leasequery [RFC7653].

## 6. Security Considerations

Security issues related with DHCPv6 are described in Section 22 of [RFC8415].

## 7. IANA Considerations

This document does not include an IANA request.

## 8. Acknowledgements

The authors would like to thank Bernie Volz, Tomek Mrugalski, Sheng Jiang, and Jinmei Tatuya for their comments and suggestions that improved the [I-D.ren-dhc-mredhcpv6]. Some ideas and thoughts of [I-D.ren-dhc-mredhcpv6] are contained in this document.

## 9. References

### 9.1. Normative References

- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.
- [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>>.

### 9.2. Informative References

- [APNA] Lee, T.L., Pappas, C.P., Barrera, D.B., Szalachowski, P.S., and A.P. Perrig, "Source Accountability with Domain-brokered Privacy", December 2016.
- [CPNR] Cisco, "Cisco Prime Network Registrar", 2018, <<https://www.cisco.com/c/en/us/products/cloud-systems-management/prime-network-registrar/index.html>>.
- [DHCP\_Broadband] Weird Solutions, "DHCP Broadband", 2018, <<https://www.weird-solutions.com/carrier-solutions/dhcp-broadband>>.

- [FreeRADIUS\_DHCP]  
FreeRADIUS, "FreeRADIUS DHCP", 2017,  
<<https://wiki.freeradius.org/features/DHCP>>.
- [GAGMS] Liu, Y.L., He, L.H., and G.R. Ren, "GAGMS: A Requirement-Driven General Address Generation and Management System", November 2017.
- [I-D.gont-v6ops-ipv6-addressing-considerations]  
Gont, F.G. and G.G. Gont, "IPv6 Addressing Considerations", February 2021.
- [I-D.jia-intarea-scenarios-problems-addressing]  
Jia, Y., Trossen, D., Iannone, L., Shenoy, N., Mendes, P., and P. Liu, "Challenging Scenarios and Problems in Internet Addressing", Work in Progress, Internet-Draft, draft-jia-intarea-scenarios-problems-addressing-02, 23 October 2021, <<https://www.ietf.org/archive/id/draft-jia-intarea-scenarios-problems-addressing-02.txt>>.
- [I-D.lhan-problems-requirements-satellite-net]  
Han, L. and R. Li, "Problems and Requirements of Satellite Constellation for Internet", Work in Progress, Internet-Draft, draft-lhan-problems-requirements-satellite-net-01, 19 October 2021, <<https://www.ietf.org/archive/id/draft-lhan-problems-requirements-satellite-net-01.txt>>.
- [I-D.ren-dhc-mredhcpv6]  
Ren, G.R., He, L.H., and Y.L. Liu, "Multi-requirement Extensions for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", March 2017.
- [ISC\_DHCP] Internet System Consortium, "ISC DHCP", 2018,  
<<http://www.isc.org/downloads/dhcp/>>.
- [Kea\_DHCP] Internet System Consortium, "Kea DHCP", 2018,  
<<https://www.isc.org/kea/>>.
- [kea\_dhcp\_hook\_developers\_guide]  
Internet Systems Consortium, "Hook Developer's Guide", 2018, <[https://jenkins.isc.org/job/Kea\\_doc/doxygen/df/d46/hooksdgDevelopersGuide.html](https://jenkins.isc.org/job/Kea_doc/doxygen/df/d46/hooksdgDevelopersGuide.html)>.
- [Microsoft]  
Microsoft, "IPv6 interface identifiers", 2013, <[https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag\\_ip\\_v6\\_imp\\_addr7.msp?mfr=true](https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sag_ip_v6_imp_addr7.msp?mfr=true)>.



- [Microsoft\_DHCP] Microsoft, "Microsoft DHCP", 2008, <[https://technet.microsoft.com/en-us/library/cc896553\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc896553(v=ws.10).aspx)>.
- [Nominum\_DHCP] Nominum, "Nominum DHCP", 2012, <[https://www.nominum.com/press\\_item/nominum-releases-new-version-of-carrier-grade-dhcp-software-for-telecom-providers/](https://www.nominum.com/press_item/nominum-releases-new-version-of-carrier-grade-dhcp-software-for-telecom-providers/)>.
- [PAVI] He, L.H., Ren, G.R., Liu, Y.L., and J.Y. Yang, "PAVI: Bootstrapping Accountability and Privacy to IPv6 Internet", April 2021.
- [RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <<https://www.rfc-editor.org/info/rfc2464>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC3898] Kalusivalingam, V., "Network Information Service (NIS) Configuration Options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3898, DOI 10.17487/RFC3898, October 2004, <<https://www.rfc-editor.org/info/rfc3898>>.
- [RFC4075] Kalusivalingam, V., "Simple Network Time Protocol (SNTP) Configuration Option for DHCPv6", RFC 4075, DOI 10.17487/RFC4075, May 2005, <<https://www.rfc-editor.org/info/rfc4075>>.
- [RFC4580] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Subscriber-ID Option", RFC 4580, DOI 10.17487/RFC4580, June 2006, <<https://www.rfc-editor.org/info/rfc4580>>.
- [RFC4649] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Remote-ID Option", RFC 4649, DOI 10.17487/RFC4649, August 2006, <<https://www.rfc-editor.org/info/rfc4649>>.

- [RFC4704] Volz, B., "The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN) Option", RFC 4704, DOI 10.17487/RFC4704, October 2006, <<https://www.rfc-editor.org/info/rfc4704>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [RFC4994] Zeng, S., Volz, B., Kinnear, K., and J. Brzozowski, "DHCPv6 Relay Agent Echo Request Option", RFC 4994, DOI 10.17487/RFC4994, September 2007, <<https://www.rfc-editor.org/info/rfc4994>>.
- [RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, DOI 10.17487/RFC5007, September 2007, <<https://www.rfc-editor.org/info/rfc5007>>.
- [RFC5460] Stapp, M., "DHCPv6 Bulk Leasequery", RFC 5460, DOI 10.17487/RFC5460, February 2009, <<https://www.rfc-editor.org/info/rfc5460>>.
- [RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6 Options for Network Boot", RFC 5970, DOI 10.17487/RFC5970, September 2010, <<https://www.rfc-editor.org/info/rfc5970>>.
- [RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options", RFC 6422, DOI 10.17487/RFC6422, December 2011, <<https://www.rfc-editor.org/info/rfc6422>>.
- [RFC6607] Kinnear, K., Johnson, R., and M. Stapp, "Virtual Subnet Selection Options for DHCPv4 and DHCPv6", RFC 6607, DOI 10.17487/RFC6607, April 2012, <<https://www.rfc-editor.org/info/rfc6607>>.
- [RFC6939] Halwasia, G., Bhandari, S., and W. Dec, "Client Link-Layer Address Option in DHCPv6", RFC 6939, DOI 10.17487/RFC6939, May 2013, <<https://www.rfc-editor.org/info/rfc6939>>.
- [RFC7037] Yeh, L. and M. Boucadair, "RADIUS Option for the DHCPv6 Relay Agent", RFC 7037, DOI 10.17487/RFC7037, October 2013, <<https://www.rfc-editor.org/info/rfc7037>>.

- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7653] Raghuvanshi, D., Kinnear, K., and D. Kukrety, "DHCPv6 Active Leasequery", RFC 7653, DOI 10.17487/RFC7653, October 2015, <<https://www.rfc-editor.org/info/rfc7653>>.
- [RFC7824] Krishnan, S., Mrugalski, T., and S. Jiang, "Privacy Considerations for DHCPv6", RFC 7824, DOI 10.17487/RFC7824, May 2016, <<https://www.rfc-editor.org/info/rfc7824>>.
- [RFC7943] Gont, F. and W. Liu, "A Method for Generating Semantically Opaque Interface Identifiers (IIDs) with the Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 7943, DOI 10.17487/RFC7943, September 2016, <<https://www.rfc-editor.org/info/rfc7943>>.
- [RFC8156] Mrugalski, T. and K. Kinnear, "DHCPv6 Failover Protocol", RFC 8156, DOI 10.17487/RFC8156, June 2017, <<https://www.rfc-editor.org/info/rfc8156>>.
- [RFC8539] Farrer, I., Sun, Q., Cui, Y., and L. Sun, "Software Provisioning Using DHCPv4 over DHCPv6", RFC 8539, DOI 10.17487/RFC8539, March 2019, <<https://www.rfc-editor.org/info/rfc8539>>.
- [VitalQIP] Nokia, "Nokia VitalQIP", 2017, <<https://networks.nokia.com/products/vitalqip-ip-address-management>>.
- [WIDE\_DHCPv6] KAME project, "WIDE DHCPv6", 2008, <[http://ipv6int.net/software/wide\\_dhcpv6.html](http://ipv6int.net/software/wide_dhcpv6.html)>.

#### Authors' Addresses

Gang Ren  
Tsinghua University  
Beijing

Phone: +86-010 6260 3227  
Email: [rengang@cernet.edu.cn](mailto:rengang@cernet.edu.cn)

Lin He  
Tsinghua University  
Beijing

Email: he-lin@tsinghua.edu.cn

Ying Liu  
Tsinghua University  
Beijing

Email: liuying@cernet.edu.cn