Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 3, 2017

R. Droms

B. Volz
Cisco Systems

O. Troan
Cisco Systems, Inc.

April 1, 2017

DHCPv6 Relay Agent Assignment Notification (RAAN) Option
draft-ietf-dhc-dhcpv6-agentopt-delegate-05.txt

Abstract

The DHCP Relay Agent Assignment Notification (RAAN) option is sent from a DHCP server to a DHCP relay agent to inform the relay agent of IPv6 addresses that have been assigned or IPv6 prefixes that have been delegated to DHCP clients.

Table of Contents

1. Introduction

The DHCP Relay Agent Assignment Notification (RAAN) option encapsulates address and prefix options to indicate that an address or prefix has been assigned. The option may also carry other information required by the network element for configuration related to the assigned address or prefix.

For example, a relay agent uses the RAAN option to learn when a prefix that has been delegated through DHCP prefix delegation (PD) to a DHCP client. The relay agent notifies the network element on which it is implemented of the delegation information so the network element can add routing information about the delegated prefix into the routing infrastructure.

While the practice to date for DHCPv6 has been for the relay agents to "snoop" the client's message (encapsulated in the received Relay Message option, and which is forwarded to the client), this will no longer be possible when clients and servers use [I-D.ietf-dhc-sedhcpv6] to encrypt their communication.

Use of the RAAN option has another benefit in that the Reply to a client's Release message, which does not have any useful information for the relay agent about the addresses or delegated prefixes the

client released, can now communicate this information in the RAAN option to the relay agent.

2. Requirements Language and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

The term "DHCP" in this document refers to DHCP for IPv6, as defined in [RFC3315]. The terms "DHCP prefix delegation" and "DHCP PD" refer DHCP for IPv6 prefix delegation, as defined in [RFC3633].

Additional terms used in the description of DHCP and DHCP prefix delegation are defined in RFC 3315 and RFC 3633. In this document "assigning" an IPv6 prefix is equivalent to "delegating" a prefix.

3. Option Semantics and Usage

The RAAN option carries information about assigned IPv6 addresses and prefixes. It encapsulates IA Address options (RFC 3315) and/or IA Prefix options (RFC 3633), and possibly other options that carry other information related to the assigned IPv6 address or prefix.

The DHCP server is responsible for synchronizing any state created by a node through the use of the RAAN option. For example, if a DHCP server receives a Release message for a delegated prefix, it causes the node to delete any state associated with that prefix by sending a RAAN option containing an IA Prefix option with the released prefix and a valid lifetime of zero.

When a DHCP server sends this option to a relay agent, it MUST include all addresses and prefixes assigned to the client on the link to which the option refers at the time the option is sent.

Examples of use:

- Populate an ACL with an assigned IPv6 address if the network security policy requires limiting IPv6 forwarding to devices that have obtained an address through DHCP.

- Inject routing information into a routing infrastructure about a delegated prefix on behalf of a requesting router.

4. Relay Agent Behavior

A relay agent that wants information from the server in a RAAN option includes an ORO requesting the RAAN option in its Relay-Forw message. A relay agent may do this for any relayed message, regardless of the message type or the message contents.

When a relay agent receives a Relay-Reply message containing a RAAN option, the relay agent may forward that option data to the node in which the relay agent is instantiated. If no RAAN option is included in the Relay-Reply, the relay agent MUST NOT assume anything with regard to RAAN data and MUST NOT forward any indication to the node in which the relay agent is instantiated.

If a node creates state based on the information included in this option, it MUST remove that state when the lifetime as specified in the option expires.

One concern with the RAAN option is that messages from the DHCP server may be received (or processed) out of order. But this concern is no different than that for the "snooping" which has been used by relay agents for many years (both in DHCPv4 and DHCPv6). Implementers should be aware of this and should consider making use of Leasequery ([RFC5007]) to resolve conflicts.

5. Server Behavior

When a server is responding to a request and the ORO contains an RAAN option, the server SHOULD include a RAAN option with all of the addresses and prefixes that have been (or are being assigned) to the client. If no addresses or prefixes are assigned, the server SHOULD send a RAAN option with no addresses or prefixes.

If the DHCP server does include this option in a Relay-Reply message, it MUST include it in the option area of the Relay-Reply message sent to the relay agent intended as the recipient of the option.

If the message received from the client contains no Client Identifier option or the server is otherwise unable to identify the client or the client’s link (perhaps because of missing or invalid data in the request), the server MUST NOT include a RAAN option in the response.

6. Option Format

The RAAN option has the following format:

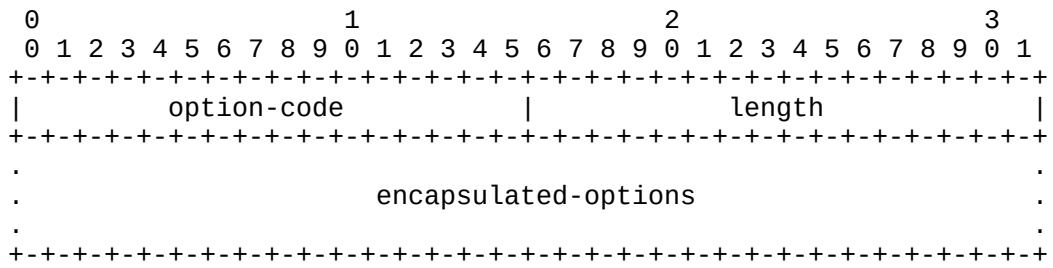


Figure 1: Relay Agent Assignment Notification Option Format

option-code OPTION_AGENT_NOTIFY (TBD).

option-len Length of encapsulated options, in octets.

encapsulated-options DHCP options to be delivered by the relay agent Assignment Notification option.

7. Encapsulating DHCP Options in the RAAN Option

The contents of options encapsulated in the RAAN option are interpreted according to the use of those options in the node on which the relay agent is implemented. For the purposes of address and prefix assignment, the uses of the DHCP IA Address and IA Prefix options are defined in this document.

Note that the contents of these options are not necessarily the same as in the corresponding options sent to the DHCP client.

7.1. IA Address Option

The fields in an IA Address option (OPTION_IAADDR, option code 5) are used as follows:

IPv6 address The IPv6 address assigned in this DHCP message

preferred-lifetime Not used by the relay agent; the server SHOULD set this field to the preferred-lifetime of the corresponding IA Address options in the message to be forwarded to the client

valid-lifetime The lifetime of the information carried in this IA Address option, expressed in units of seconds; if the valid-lifetime is 0, the information is no longer valid

IAaddr-options	Not used by the relay agent; the server SHOULD set this field to the IAaddr-options of the corresponding IA Address option in the message to be forwarded to the client

7.2. IA Prefix Option

The fields in an IA Prefix option (OPTION_IAPREFIX, option code 28) are used as follows:

preferred-lifetime	Not used by the relay agent; the server SHOULD set this field to the preferred-lifetime of the corresponding IA Prefix options in the message to be forwarded to the client
valid-lifetime	The lifetime of the information carried in this IA Prefix option, expressed in units of seconds; if the valid-lifetime is 0, the information is no longer valid
prefix-length	Length for this prefix in bits
IPv6-prefix	The IPv6 prefix assigned in this DHCP message
IAprefix-options	Not used by the relay agent; the server SHOULD set this field to the IAprefix-options of the corresponding IA Prefix option in the message to be forwarded to the client

8. Requesting Assignment Information from the DHCP Server

If a relay agent requires the DHCP server to provide information about assigned addresses and prefixes, it MUST include an Option Request option, requesting the Assignment Notification option, as described in section 22.7 of RFC 3315.

9. IANA Considerations

IANA is requested to assign an option code from the "DHCPv6 and DHCPv6 options" registry http://www.iana.org/assignments/dhcpv6-parameters to OPTION_AGENT_NOTIFY.

10. Security Considerations

Security issues related to DHCP are described in RFC 3315 and RFC 3633.

The RAAN option may be used to mount a denial of service attack by causing a node to incorrectly populate an ACL or incorrectly configure routing information for a delegated prefix. This option may also be used to insert invalid prefixes into the routing infrastructure or add invalid IP addresses to ACLs in nodes. Communication between a server and a relay agent, and communication between relay agents, can be secured through the use of IPSec, as described in [I-D.ietf-dhc-relay-server-security].

11. Changes Log

If this section is included in the document when it is submitted for publication, the RFC Editor is requested to remove it.

Changes in rev -01:

- Added section describing use of "Server Reply Sequence Number" option to allow resequencing of out-of-order messages.

Changes in rev -02:

- Made editorial change in section 1: s/the appropriate routing protocols/the routing infrastructure/
- Updated first paragraph in Section 3 to allow multiple IA Address options and/or IA Prefix options
- Renamed section 3 to "Options Semantics and Usage"
- Added paragraph to section "Option Semantics and Usage" requiring that the DHCP server must include all addresses/ prefixes for the client (on that link) in the RAAN option
- Added list of use cases to section "Option Semantics and Usage"
- Added section "Relay Agent Behavior"
- Added section "Server Behavior"; moved second paragraph of section "Option Semantics and Usage" to "Server Behavior"
- Updated reference to draft-ietf-dhc-dhcpv6-srsn-option-00
- Clarified descriptions of various option fields in section "Encapsulating DHCP options in the RAAN Option"

Changes in rev -03: refreshed after expiration.

Changes in rev -04: all references to the "Server Reply Sequence Number" option were removed from the draft.

Changes in rev -05:

- Converted the -04 text version to xml.
- Updated introduction to add motivation for option because of [I-D.ietf-dhc-sedhcpv6], and also Reply to Release "snooping" issues.
- Updated security considerations to reference IPsec document ([I-D.ietf-dhc-relay-server-security]).

12. References

12.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

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

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

12.2. Informative References

[I-D.ietf-dhc-relay-server-security] Volz, B. and Y. Pal, "Security of Messages Exchanged Between Servers and Relay Agents", draft-ietf-dhc-relay-server-security-04 (work in progress), March 2017.

[I-D.ietf-dhc-sedhcpv6] Li, L., Jiang, S., Cui, Y., Jinmei, T., Lemon, T., and D. Zhang, "Secure DHCPv6", draft-ietf-dhc-sedhcpv6-21 (work in progress), February 2017.

[RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, DOI 10.17487/RFC5007, September 2007, <http://www.rfc-editor.org/info/rfc5007>.

Authors' Addresses

Ralph Droms

Email: rdroms.ietf@gmail.com

Bernie Volz
Cisco Systems, Inc.
1414 Massachusetts Ave
Boxborough, MA 01719
USA

Email: volz@cisco.com

Ole Troan
Cisco Systems, Inc.
Oslo
Norway

Email: otroan@cisco.com

DHC Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 25, 2017

L. Li
Tsinghua University
S. Jiang
Huawei Technologies Co., Ltd
Y. Cui
Tsinghua University
T. Jinmei
Infoblox Inc.
T. Lemon
Nominum, Inc.
D. Zhang
February 21, 2017

Secure DHCPv6
draft-ietf-dhc-sedhcpv6-21

Abstract

DHCPv6 includes no deployable security mechanism that can protect end-to-end communication between DHCP clients and servers. This document describes a mechanism for using public key cryptography to provide such security. The mechanism provides encryption in all cases, and can be used for authentication based on pre-sharing of authorized certificates.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 25, 2017.

Copyright Notice

Table of Contents

1. Introduction

The Dynamic Host Configuration Protocol for IPv6 (DHCPv6, [RFC3315]) allows DHCPv6 servers to flexibly provide addressing and other configuration information relating to local network infrastructure to DHCP clients. The protocol provides no deployable security mechanism, and consequently is vulnerable to various attacks.

This document provides a brief summary of the security vulnerabilities of the DHCPv6 protocol and then describes a new extension to the protocol that provides two additional types of security:

- o authentication of the DHCPv6 client and the DHCPv6 server to defend against active attacks, such as spoofing.

- o encryption between the DHCPv6 client and the DHCPv6 server in order to protect the DHCPv6 communication from pervasive monitoring.

The extension specified in this document applies only to end-to-end communication between DHCP servers and clients. Options added by relay agents in Relay-Forward messages, and options other than the client message in Relay-Reply messages sent by DHCP servers, are not protected. Such communications are already protected using the mechanism described in [I-D.ietf-dhc-relay-server-security].

This extension introduces two new DHCPv6 messages: the Encrypted-Query and the Encrypted-Response messages. It defines six new DHCPv6 options: the Algorithm, Certificate, Signature, Increasing-number, Encryption-Key-Tag option and Encrypted-message options.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

3. Terminology

This section defines terminology specific to secure DHCPv6 used in this document.

secure DHCPv6 client: A node that initiates a DHCPv6 request on a link to obtain DHCPv6 configuration parameters from

one or more DHCPv6 servers using the encryption and optional authentication mechanisms defined in this document.

secure DHCPv6 server: A DHCPv6 server that implements the authentication and encryption mechanisms defined in this document, and is configured to use them.

4. Security Issues of DHCPv6

[RFC3315] defines an authentication mechanism with integrity protection. This mechanism uses a symmetric key that is shared by the client and server for authentication. It does not provide any key distribution mechanism.

For this approach, operators can set up a key database for both servers and clients from which the client obtains a key before running DHCPv6. However, manual key distribution runs counter to the goal of minimizing the configuration data needed at each host. Consequently, there are no known deployments of this security mechanism.

[RFC3315] provides an additional mechanism for preventing off-network timing attacks using the Reconfigure message: the Reconfigure Key authentication method. However, this method protects only the Reconfigure message. The key is transmitted in plaintext to the client in earlier exchanges and so this method is vulnerable to on-path active attacks.

Anonymity Profile for DHCP Clients [RFC7844] explains how to generate DHCPv4 or DHCPv6 requests that minimize the disclosure of identifying information. However, the anonymity profile limits the use of the certain options. It also cannot anticipate new options that may contain private information. In addition, the anonymity profile does not work in cases where the client wants to maintain anonymity from eavesdroppers but must identify itself to the DHCP server with which it intends to communicate.

Privacy consideration for DHCPv6 [RFC7824] presents an analysis of the privacy issues associated with the use of DHCPv6 by Internet users. No solutions are presented.

Current DHCPv6 messages are still transmitted in cleartext and the privacy information within the DHCPv6 message is not protected from passive attack, such as pervasive monitoring [RFC7258]. The privacy information of the IPv6 host, such as DUID, may be gleaned to find location information, previous visited networks and so on. [RFC7258]

claims that pervasive monitoring should be mitigated in the design of IETF protocol, where possible.

To better address the problem of passive monitoring and to achieve authentication without requiring a symmetric key distribution solution for DHCP, this document defines an asymmetric key authentication and encryption mechanism. This protects against both active attacks, such as spoofing, and passive attacks, such as pervasive monitoring.

5. Secure DHCPv6 Overview

5.1. Solution Overview

The following figure illustrates the secure DHCPv6 procedure. Briefly, this extension establishes the server's identity with an anonymous Information-Request exchange. Once the server's identity has been established, the client may either choose to communicate with the server or not. Not communicating with an unknown server avoids revealing private information, but if there is no known server on a particular link, the client will be unable to communicate with a DHCP server.

If the client chooses to communicate with the selected server(s), it uses the Encrypted-Query message to encapsulate its communications to the DHCP server. The server responds with Encrypted-Response messages. Normal DHCP messages are encapsulated in these two new messages using the new defined Encrypted-message option. Besides the Encrypted-message option, the Signature option is defined to verify the integrity of the DHCPv6 messages and then authentication of the client and the server. The Increasing number option is defined to detect a replay attack.

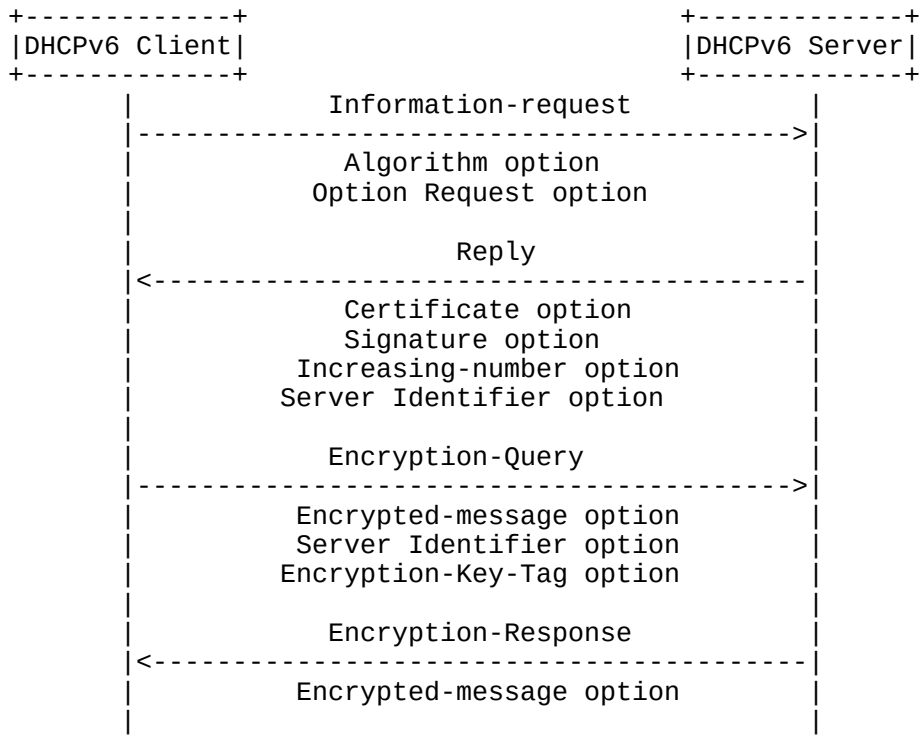


Figure 1: Secure DHCPv6 Procedure

5.2. New Components

The new components of the mechanism specified in this document are as follows:

- Servers and clients that use certificates first generate a public/private key pair and then obtain a certificate that signs the public key. The Certificate option is defined to carry the certificate of the sender.

- The algorithm option is defined to carry the algorithms lists for algorithm agility.

- The signature is generated using the private key to verify the integrity of the DHCPv6 messages. The Signature option is defined to carry the signature.

- The increasing number is used to detect replayed packet. The Increasing-number option is defined to carry a strictly-increasing serial number.

- The encryption key Tag is calculated from the public key data. The Encryption-Key-Tag option is defined to identify the used public/private key pair.

- The Encrypted-message option is defined to contain the encrypted DHCPv6 message.

- The Encrypted-Query message is sent from the secure DHCPv6 client to the secure DHCPv6 server. The Encrypted-Query message MUST contain the Encrypted-message option and Encryption-Key-Tag option. In addition, the Server Identifier option MUST be included if it is contained in the original DHCPv6 message. The Encrypted-Query message MUST NOT contain any other options.

- The Encrypted-Response message is sent from the secure DHCPv6 server to the secure DHCPv6 client. The Encrypted-Response message MUST contain the Encrypted-message option. The Encrypted-Response message MUST NOT contain any other options.

5.3. Support for Algorithm Agility

In order to provide a means of addressing problems that may emerge with existing hash algorithms, signature algorithm and encryption algorithms in the future, this document provides a mechanism to support algorithm agility. The support for algorithm agility in this document is mainly a algorithm notification mechanism between the client and the server. The same client and server MUST use the same algorithm in a single communication session. The client can offer a set of algorithms, and then the server selects one algorithm for the future communication.

5.4. Impact on RFC3315

For secure DHCPv6, the Solicit and Rebind messages can be sent only to the selected server(s) which share one common certificate. If the client doesn’t like the received Advertise(s) it could restart the whole process and selects another certificate, but it will be more expensive, and there’s no guarantee that other servers can provide better Advertise(s).

[RFC3315] provides an additional mechanism for preventing off-network timing attacks using the Reconfigure message: the Reconfigure Key authentication method. Secure DHCPv6 can protect the Reconfigure message using the encryption method. So the Reconfigure Key authentication method SHOULD NOT be used if Secure DHCPv6 is applied.

5.5. Applicability

In principle, secure DHCPv6 is applicable in any environment where physical security on the link is not assured and attacks on DHCPv6 are a concern. In practice, however, authenticated and encrypted DHCPv6 configuration will rely on some operational assumptions mainly regarding public key distribution and management. In order to achieve the wider use of secure DHCPv6, opportunistic security [RFC7435] can be applied to secure DHCPv6 deployment, which allows DHCPv6 encryption in environments where support for authentication or a key distribution mechanism is not available.

Secure DHCPv6 can achieve authentication and encryption based on pre-sharing of authorized certificates. One feasible environment in an early deployment stage would be enterprise networks. In enterprise networks, the client is manually pre-configured with the trusted servers' public key and the server can also be manually pre-configured with the trusted clients' public keys. In some scenario, such as coffee shop where the certificate cannot be validated and one wants access to the Internet, then the DHCPv6 configuration process can be encrypted without authentication.

Note that this deployment scenario based on manual operation is not much different from the existing, shared-secret based authentication mechanisms defined in [RFC3315] in terms of operational costs. However, Secure DHCPv6 is still securer than the shared-secret mechanism in that even if clients' keys stored for the server are stolen that does not mean an immediate threat as these are public keys. In addition, if some kind of Public Key Infrastructure (PKI) is used with Secure DHCPv6, even if the initial installation of the certificates is done manually, it will help reduce operational costs of revocation in case a private key (especially that of the server) is compromised.

6. DHCPv6 Client Behavior

The secure DHCPv6 client is pre-configured with a certificate and its corresponding private key for client authentication. If the client does not obtain a certificate from Certificate Authority (CA), it can generate the self-signed certificate.

The secure DHCPv6 client sends an Information-request message as per [RFC3315]. The Information-request message is used by the DHCPv6 client to request the server's certificate information without having addresses, prefixes or any non-security options assigned to it. The contained Option Request option MUST carry the option code of the Certificate option. In addition, the contained Algorithm option MUST be constructed as explained in Section 10.1.1. The Information-

request message MUST NOT include any other DHCPv6 options except the above options to minimize the client's privacy information leakage.

When receiving the Reply messages from the DHCPv6 servers, a secure DHCPv6 client discards any DHCPv6 message that meets any of the following conditions:

- the Signature option is missing,
- multiple Signature options are present,
- the Certificate option is missing.

And then the client first checks acknowledged hash, signature and encryption algorithms that the server supports. The client checks the signature/encryption algorithms through the certificate option and checks the signature/hash algorithms through the signature option. The SA-id in the certificate option must be equal to the SA-id in the signature option. If they are different, then the client drops the Reply message. The client uses the acknowledged algorithms in the subsequent messages.

Then the client checks the authority of the server. In some scenario where non-authenticated encryption can be accepted, such as coffee shop, then authentication is optional and can be skipped. For the certificate check method, the client validates the certificates through the pre-configured local trusted certificates list or other methods. A certificate that finds a match in the local trust certificates list is treated as verified. If the certificate check fails, the Reply message is dropped.

The client MUST now authenticate the server by verifying the signature and checking increasing number, if there is a Increasing-number option. The order of two procedures is left as an implementation decision. It is RECOMMENDED to check increasing number first, because signature verification is much more computationally expensive. The client checks the Increasing-number option according to the rule defined in Section 9.1. For the message without an Increasing-number option, according to the client's local policy, it MAY be acceptable or rejected. The Signature field verification MUST show that the signature has been calculated as specified in Section 10.1.3. Only the messages that get through both the signature verification and increasing number check (if there is a Increasing-number option) are accepted. Reply message that does not pass the above tests MUST be discarded.

If there are multiple authenticated DHCPv6 certs, the client selects one DHCPv6 cert for the following communication. The selected

certificate may correspond to multiple DHCPv6 servers. If there are no authenticated DHCPv6 certs or existing servers fail authentication, the client should retry a number of times. The client conducts the server discovery process as per section 18.1.5 of [RFC3315] to avoid a packet storm. In this way, it is difficult for a rogue server to beat out a busy "real" server. And then the client takes some alternative action depending on its local policy, such as attempting to use an unsecured DHCPv6 server.

Once the server has been authenticated, the DHCPv6 client sends the Encrypted-Query message to the DHCPv6 server. The Encrypted-Query message contains the Encrypted-message option, which MUST be constructed as explained in Section 10.1.6. The Encrypted-message option contains the encrypted DHCPv6 message using the public key contained in the selected cert. In addition, the Server Identifier option MUST be included if it is in the original message (i.e. Request, Renew, Decline, Release) to avoid the need for other servers receiving the message to attempt to decrypt it. The Encrypted-Query message MUST include the Encryption-Key-Tag option to identify the used public/private key pair, which is constructed as explained in Section 10.1.5. The Encrypted-Query message MUST NOT contain any other DHCPv6 option except the Server Identifier option, Encryption-Key-Tag option, Encrypted-Message option.

The first DHCPv6 message sent from the client to the server, such as Solicit message, MUST contain the related information for client authentication. The encryption text SHOULD be formatted as explain in [RFC5652]. The Certificate option MUST be constructed as explained in Section 10.1.2. In addition, one and only one Signature option MUST be contained, which MUST be constructed as explained in Section 10.1.3. One and only one Increasing-number option SHOULD be contained, which MUST be constructed as explained in Section 10.1.4. In addition, the subsequent encrypted DHCPv6 message sent from the client can also contain the Increasing-number option to defend against replay attack.

For the received Encrypted-Response message, the client MUST drop the Encrypted-Response message if other DHCPv6 option except Encrypted-message option is contained. If the transaction-id is 0, the client also try to decrypt it. Then, the client extracts the Encrypted-message option and decrypts it using its private key to obtain the original DHCPv6 message. In this document, it is assumed that the client will not have multiple DHCPv6 sessions with different DHCPv6 servers using different key pairs and only one key pair is used for the encrypted DHCPv6 configuration process. After the decryption, it handles the message as per [RFC3315].If the decrypted DHCPv6 message contains the Increasing-number option, the DHCPv6 client checks it according to the rule defined in Section 9.1.

If the client fails to get the proper parameters from the chosen server(s), it can select another authenticated certificate and send the Encrypted-Query message to another authenticated server(s) for parameters configuration until the client obtains the proper parameters.

When the decrypted message is Reply message with an error status code, the error status code indicates the failure reason on the server side. According to the received status code, the client MAY take follow-up action:

- Upon receiving an AuthenticationFail error status code, the client is not able to build up the secure communication with the server. However, there may be other DHCPv6 servers available that successfully complete authentication. The client MAY use the AuthenticationFail as a hint and switch to other DHCPv6 server if it has another one. The client SHOULD retry with another authenticated certificate. However, if the client decides to retransmit using the same certificate after receiving AuthenticationFail, it MUST NOT retransmit immediately and MUST follow normal retransmission routines defined in [RFC3315].

- Upon receiving a ReplayDetected error status code, the client MAY resend the message with an adjusted Increasing-number option according to the returned number from the DHCPv6 server.

- Upon receiving a SignatureFail error status code, the client MAY resend the message following normal retransmission routines defined in [RFC3315].

7. DHCPv6 Server Behavior

The secure DHCPv6 server is pre-configured with a certificate and its corresponding private key for server authentication. If the server does not obtain the certificate from Certificate Authority (CA), it can generate the self-signed certificate.

When the DHCPv6 server receives the Information-request message and the contained Option Request option identifies the request is for the server’s certificate information, it SHOULD first check the hash, signature, encryption algorithms sets that the client supports. The server selects one hash, signature, encryption algorithm from the acknowledged algorithms sets for the future communication. And then, the server replies with a Reply message to the client. The Reply message MUST contain the requested Certificate option, which MUST be constructed as explained in Section 10.1.2, and Server Identifier option. In addition, the Reply message MUST contain one and only one Signature option, which MUST be constructed as explained in

Section 10.1.3. Besides, the Reply message SHOULD contain one and only one Increasing-number option, which MUST be constructed as explained in Section 10.1.4.

Upon the receipt of Encrypted-Query message, the server MUST drop the message if the other DHCPv6 option is contained except Server Identifier option, Encryption-Key-Tag option, Encrypted-message option. Then, the server checks the Server Identifier option. The DHCPv6 server drops the message that is not for it, thus not paying cost to decrypt messages. If it is the target server, according to the Encryption-Key-Tag option, the server identifies the used public/private key pair and decrypts the Encrypted-message option using the corresponding private key. It is essential to note that the encryption key tag is not a unique identifier. It is theoretically possible for two different public keys to share one common encryption key tag. The encryption key tag is used to limit the possible candidate keys, but it does not uniquely identify a public/private key pair. The server MUST try all corresponding key pairs. If the server cannot find the corresponding private key of the key tag or the corresponding private key of the key tag is invalid for decryption, then the server drops the received message.

If secure DHCPv6 server needs client authentication and decrypted message is a Solicit/Information-request message which contains the information for client authentication, the secure DHCPv6 server discards the received message that meets any of the following conditions:

- the Signature option is missing,
- multiple Signature options are present,
- the Certificate option is missing.

For the signature failure, the server SHOULD send an encrypted Reply message with an UnspecFail (value 1, [RFC3315]) error status code to the client.

The server validates the client’s certificate through the local pre-configured trusted certificates list. A certificate that finds a match in the local trust certificates list is treated as verified. If the server does not know the certificate and can accept the non-authenticated encryption, then the server skips the authentication process and uses it for encryption only. The message that fails authentication validation MUST be dropped. In such failure, the DHCPv6 server replies with an encrypted Reply message with an AuthenticationFail error status code, defined in Section 10.3, back

to the client. At this point, the server has either recognized the authentication of the client, or decided to drop the message.

If the decrypted message contains the Increasing-number option, the server checks it according to the rule defined in Section 9.1. If the check fails, an encrypted Reply message with a ReplayDetected error status code, defined in Section 10.3, should be sent back to the client. In the Reply message, a Increasing-number option is carried to indicate the server's stored number for the client to use. According to the server's local policy, the message without an Increasing-number option MAY be acceptable or rejected.

The Signature field verification MUST show that the signature has been calculated as specified in Section 10.1.3. If the signature check fails, the DHCPv6 server SHOULD send an encrypted Reply message with a SignatureFail error status code. Only the clients that get through both the signature verification and increasing number check (if there is a Increasing-number option) are accepted as authenticated clients and continue to be handled their message as defined in [RFC3315].

Once the client has been authenticated, the DHCPv6 server sends the Encrypted-response message to the DHCPv6 client. If the DHCPv6 message is Reconfigure message, then the server set the transaction-id of the Encrypted-Response message to 0. The Encrypted-response message MUST only contain the Encrypted-message option, which MUST be constructed as explained in Section 10.1.6. The encryption text SHOULD be formatted as explain in [RFC5652]. The Encrypted-message option contains the encrypted DHCPv6 message that is encrypted using the authenticated client's public key. To provide the replay protection, the Increasing-number option SHOULD be contained in the encrypted DHCPv6 message.

8. Relay Agent Behavior

When a DHCPv6 relay agent receives an Encrypted-query or Encrypted-response message, it may not recognize this message. The unknown messages MUST be forwarded as described in [RFC7283].

When a DHCPv6 relay agent recognizes the Encrypted-query and Encrypted-response messages, it forwards the message according to section 20 of [RFC3315]. There is nothing more the relay agents have to do, it neither needs to verify the messages from client or server, nor add any secure DHCPv6 options. Actually, by definition in this document, relay agents MUST NOT add any secure DHCPv6 options.

Relay-forward and Relay-reply messages MUST NOT contain any additional Certificate option or Increasing-number option, aside from

those present in the innermost encapsulated messages from the client or server.

9. Processing Rules

9.1. Increasing Number Check

In order to check the Increasing-number option, defined in Section 10.1.4, the client/server has one stable stored number for replay attack detection. The server should keep a record of the increasing number forever. And the client keeps a record of the increasing number during the DHCPv6 configuration process with the DHCPv6 server. And the client can forget the increasing number information after the transaction is finished. The client's initial locally stored increasing number is set to zero.

It is essential to remember that the increasing number is finite. All arithmetic dealing with sequence numbers must be performed modulo 2^64. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2^64 - 1 to 0 again.

In order to check the Increasing-number option, the following comparison is needed.

NUM.STO = the stored number in the client/server

NUM.REC = the acknowledged number from the received message

The Increasing-number option in the received message passes the increasing number check if NUM.REC is more than NUM.STO. And then, the value of NUM.STO is changed into the value of NUM.REC.

The increasing number check fails if NUM.REC is equal with or less than NUM.STO.

9.2. Encryption Key Tag Calculation

The generation method of the encryption key tag adopts the method define in Appendix B in [RFC4034].

The following reference implementation calculates the value of the encryption key tag. The input is the data of the public key. The code is written for clarity not efficiency.

```
/*
    * First octet of the key tag is the most significant 8 bits of the
    * return value;
    * Second octet of the key tag is the least significant 8 bits of the
    * return value.
*/

   unsigned int
   keytag (
           unsigned char key[],  /* the RDATA part of the DNSKEY RR */
           unsigned int keysize  /* the RDLENGTH */
          )
   {
           unsigned long ac;     /* assumed to be 32 bits or larger */
           int i;                /* loop index */

           for ( ac = 0, i = 0; i < keysize; ++i )
                   ac += (i & 1) ? key[i] : key[i] << 8;
           ac += (ac >> 16) & 0xFFFF;
           return ac & 0xFFFF;
   }
```

10. Extensions for Secure DHCPv6

This section describes the extensions to DHCPv6. Six new DHCPv6 options, two new DHCPv6 messages and six new status codes are defined.

10.1. New DHCPv6 Options

10.1.1. Algorithm Option

The Algorithm option carries the algorithms sets for algorithm agility, which is contained in the Information-request message.

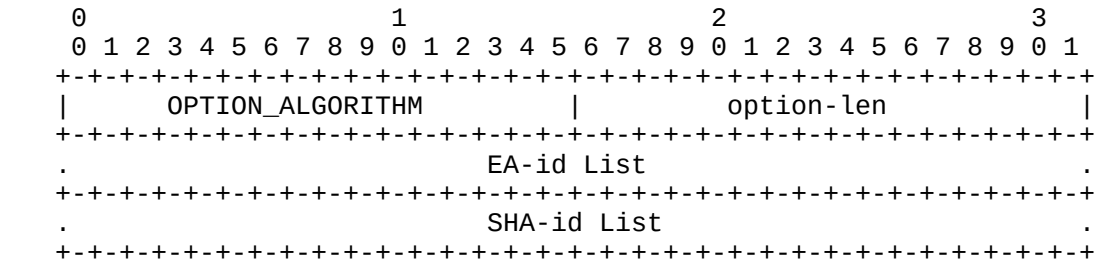
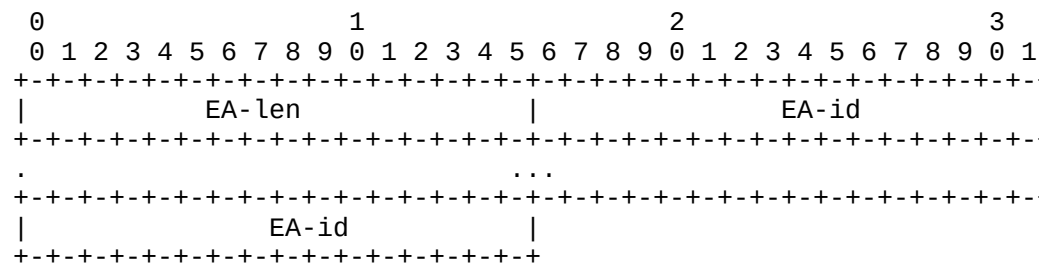


Figure 2: Algorithm Option

   o option-code: OPTION_ALGORITHM (TBA1).

   o option-len: length of EA-id List + length of SHA-id List in octets.

   o EA-id: The format of the EA-id List field is shown in Figure 3.

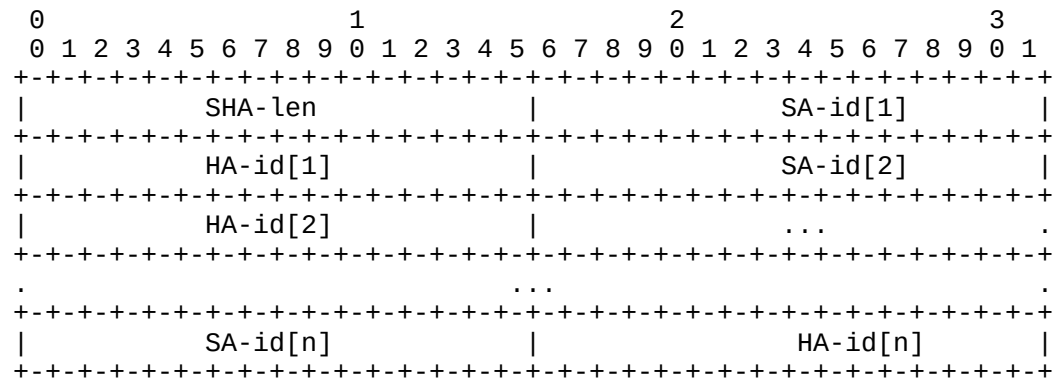


EA-len The length of the following EA-ids.

EA-id 2-octets value to indicate the Encryption Algorithm id. The client enumerates the list of encryption algorithms it supports to the server. The encryption algorithm is used for the encrypted DHCPv6 configuration process. This design is adopted in order to provide encryption algorithm agility. The value is from the Encryption Algorithm for Secure DHCPv6 registry in IANA. A registry of the initial assigned values is defined in Section 12. The RSA algorithm, as the mandatory encryption algorithm, MUST be included.

Figure 3: EA-id List Field

   o SHA-id List: The format of the SHA-id List field is shown in Figure 4. The SHA-id List contains multiple pair of (SA-id, HA-id). Each pair of (SA-id[i], HA-id[i]) is considered to specify a specific signature method.



SHA-len The length of the following SA-id and HA-id pairs.

SA-id 2-octets value to indicate the Signature Algorithm id. The client enumerates the list of signature algorithms it supports to the server. This design is adopted in order to provide signature algorithm agility. The value is from the Signature Algorithm for Secure DHCPv6 registry in IANA. The support of RSASSA-PKCS1-v1_5 is mandatory. A registry of the initial assigned values is defined in Section 12. The mandatory signature algorithms MUST be included.

HA-id 2-octets value to indicate the Hash Algorithm id. The client enumerates the list of hash algorithms it supports to the server. This design is adopted in order to provide hash algorithm agility. The value is from the Hash Algorithm for Secure DHCPv6 registry in IANA. The support of SHA-256 is mandatory. A registry of the initial assigned values is defined in Section 12. The mandatory hash algorithms MUST be included.

Figure 4: SHA-id List Field

10.1.2. Certificate Option

The Certificate option carries the certificate of the client/server, which is contained in the Reply message. The format of the Certificate option is described as follows:

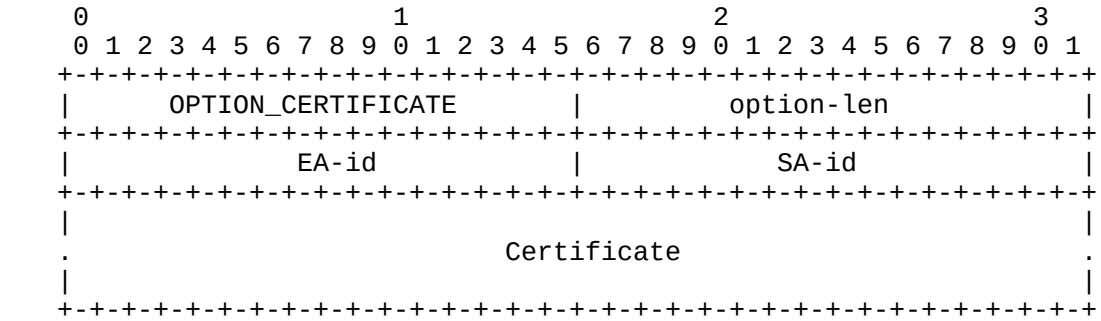


Figure 5: Certificate Option

- option-code: OPTION_CERTIFICATE (TBA2).
- option-len: 4 + length of Certificate in octets.
- EA-id: Encryption Algorithm id which is used for the certificate. If the value of the EA-id is 0, then the public key in the certificate is not used for encryption calculation.
- SA-id: Signature Algorithm id which is used for the certificate. If the value of the EA-id is 0, then the public key in the certificate is not used for signature calculation.
- Certificate: A variable-length field containing certificates. The encoding of certificate and certificate data MUST be in format as defined in Section 3.6, [RFC7296]. The support of X.509 certificate is mandatory.

It should be noticed that the scenario where the values of EA-id and SA-id are both 0 makes no sense and the client MUST discard a message with such values.

10.1.3. Signature option

The Signature option contains a signature that is signed by the private key to be attached to the Reply message. The Signature option could be in any place within the DHCPv6 message while it is logically created after the entire DHCPv6 header and options. It protects the entire DHCPv6 header and options, including itself. The format of the Signature option is described as follows:

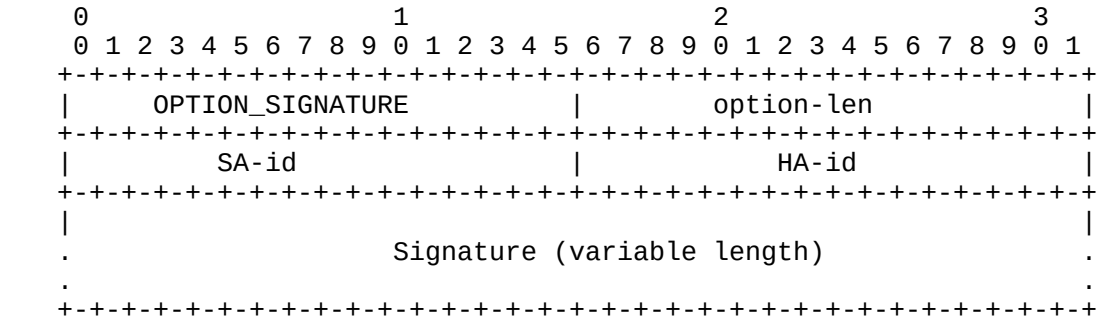


Figure 6: Signature Option

- option-code: OPTION_SIGNATURE (TBA3).

- option-len: 4 + length of Signature field in octets.

- SA-id: Signature Algorithm id. The signature algorithm is used for computing the signature result. This design is adopted in order to provide signature algorithm agility. The value is from the Signature Algorithm for Secure DHCPv6 registry in IANA. The support of RSASSA-PKCS1-v1_5 is mandatory. A registry of the initial assigned values is defined in Section 12.

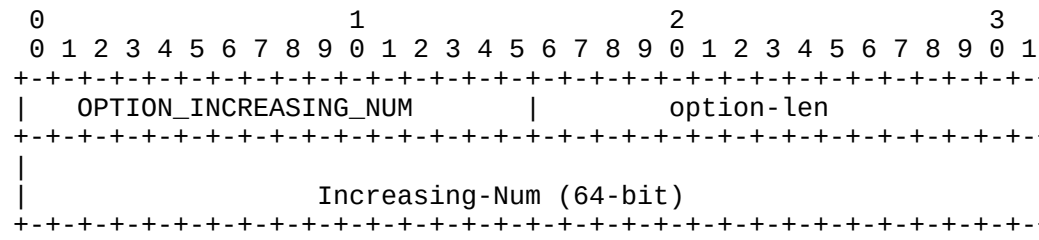
- HA-id: Hash Algorithm id. The hash algorithm is used for computing the signature result. This design is adopted in order to provide hash algorithm agility. The value is from the Hash Algorithm for Secure DHCPv6 registry in IANA. The support of SHA-256 is mandatory. A registry of the initial assigned values is defined in Section 12.

- Signature: A variable-length field containing a digital signature. The signature value is computed with the hash algorithm and the signature algorithm, as described in HA-id and SA-id. The Signature field MUST be padded, with all 0, to the next octet boundary if its size is not a multiple of 8 bits. The padding length depends on the signature algorithm, which is indicated in the SA-id field.

Note: If Secure DHCPv6 is used, the DHCPv6 message is encrypted in a way that the authentication mechanism defined in RFC3315 does not understand. So the Authentication option SHOULD NOT be used if Secure DHCPv6 is applied.

10.1.4. Increasing-number Option

The Increasing-number option carries the strictly increasing number for anti-replay protection, which is contained in the Reply message and the encrypted DHCPv6 message. It is optional.



option-code OPTION_INCREASING_NUM (TBA4).

option-len 8, in octets.

Increasing-Num A strictly increasing number for the replay attack detection which is more than the local stored number.

Figure 7: Increasing-number Option

10.1.5. Encryption-Key-Tag Option

The Encryption-Key-Tag option carries the key identifier which is calculated from the public key data. The Encrypted-Query message MUST contain the Encryption-Key-Tag option to identify the used public/private key pair.

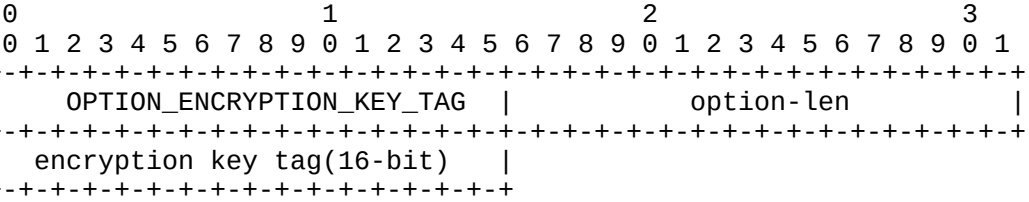


Figure 8: Encryption-Key-Tag Option

option-code OPTION_ENCRYPTION_KEY_TAG (TBA5).

option-len 2, in octets.

encryption key tag A 16 bits field containing the encryption key tag sent from the client to server to identify the used public/private key pair. The encryption key tag is calculated from the public

key data, like fingerprint of a specific public key. The specific calculation method of the encryption key tag is illustrated in Section 9.2.

10.1.6. Encrypted-message Option

The Encrypted-message option carries the encrypted DHCPv6 message, which is calculated with the recipient's public key. The Encrypted-message option is contained in the Encrypted-Query message or the Encrypted-Response message.

The format of the Encrypted-message option is:

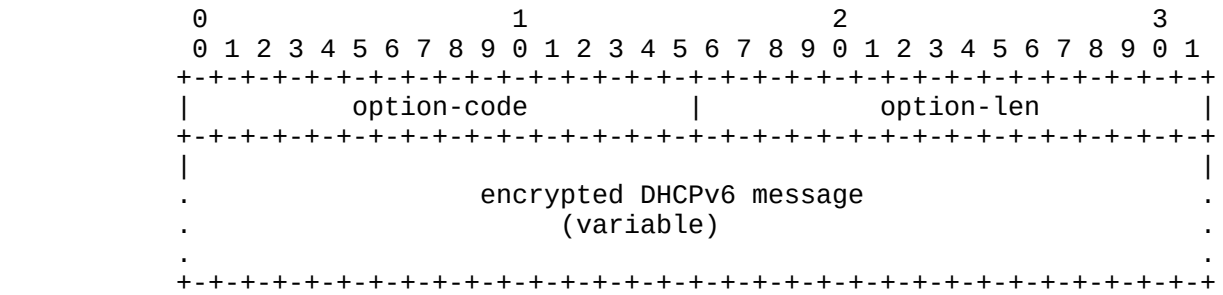


Figure 9: Encrypted-message Option

option-code OPTION_ENCRYPTED_MSG (TBA6).

option-len Length of the encrypted DHCPv6 message in octets.

encrypted DHCPv6 message A variable length field containing the encrypted DHCPv6 message. In Encrypted-Query message, it contains encrypted DHCPv6 message sent from a client to server. In Encrypted-response message, it contains encrypted DHCPv6 message sent from a server to client.

10.2. New DHCPv6 Messages

Two new DHCPv6 messages are defined to achieve the DHCPv6 encryption: Encrypted-Query and Encrypted-Response. Both the DHCPv6 messages defined in this document share the following format:

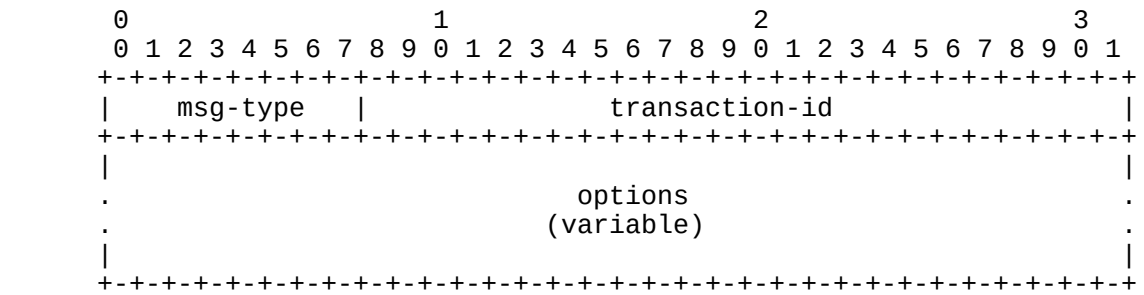


Figure 10: The format of Encrypted-Query and Encrypted-Response Messages

msg-type Identifier of the message type. It can be either Encrypted-Query (TBA7) or DHCPv6-Response (TBA8).

transaction-id The transaction ID for this message exchange.

options The Encrypted-Query message MUST contain the Encrypted-message option, Encryption-Key-Tag option and Server Identifier option if the message in the Encrypted-message option has a Server Identifier option. The Encrypted-Response message MUST only contain the Encrypted-message option.

10.3. Status Codes

The following new status codes, see Section 5.4 of [RFC3315] are defined.

- AuthenticationFail (TBD9): indicates that the message from the DHCPv6 client fails authentication check.
- ReplayDetected (TBD10): indicates the message from DHCPv6 client fails the increasing number check.
- SignatureFail (TBD11): indicates the message from DHCPv6 client fails the signature check.

11. Security Considerations

This document provides the authentication and encryption mechanisms for DHCPv6.

There are some mandatory algorithm for encryption algorithm in this document. It may be at some point that the mandatory algorithm is no longer safe to use.

A server or a client, whose local policy accepts messages without a Increasing-number option, may have to face the risk of replay attacks.

Since the algorithm option isn't protected by a signature, the list can be forged without detection, which can lead to a downgrade attack.

Likewise, since the Encryption-Key-Tag Option isn't protected, an attacker that can intercept the message can forge the value without detection.

If the client tries more than one cert for client authentication, the server can easily get a client that implements this to enumerate its entire cert list and probably learn a lot about a client that way. For this security item, It is RECOMMENDED that client certificates could be tied to specific server certificates by configuration.

12. IANA Considerations

This document defines six new DHCPv6 [RFC3315] options. The IANA is requested to assign values for these six options from the DHCPv6 Option Codes table of the DHCPv6 Parameters registry maintained in http://www.iana.org/assignments/dhcpv6-parameters. The six options are:

The Algorithm Option (TBA1), described in Section 10.1.2.

The Certificate Option (TBA2), described in Section 10.1.2.

The Signature Option (TBA3), described in Section 10.1.3.

The Increasing-number Option (TBA4),described in Section 10.1.4.

The Encryption-Key-Tag Option (TBA5),described in Section 10.1.5.

The Encrypted-message Option (TBA6), described in Section 10.1.6.

The IANA is also requested to assign value for these two messages from the DHCPv6 Message Types table of the DHCPv6 Parameters registry maintained in http://www.iana.org/assignments/dhcpv6-parameters. The two messages are:

The Encrypted-Query Message (TBA7), described in Section 10.2.

The Encrypted-Response Message (TBA8), described in Section 10.2.

The IANA is also requested to add three new registry tables to the DHCPv6 Parameters registry maintained in http://www.iana.org/assignments/dhcpv6-parameters. The three tables are the Hash Algorithm for Secure DHCPv6 table, the Signature Algorithm for Secure DHCPv6 table and the Encryption Algorithm for Secure DHCPv6 table.

Initial values for these registries are given below. Future assignments are to be made through Standards Action [RFC5226]. Assignments for each registry consist of a name, a value and a RFC number where the registry is defined.

Hash Algorithm for Secure DHCPv6. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Hash Algorithm for Secure DHCPv6 in this document:

Name	Value	RFCs
SHA-256	0x01	this document
SHA-512	0x02	this document

Signature Algorithm for Secure DHCPv6. The values in this table are 16-bit unsigned integers. The following initial values are assigned for Signature Algorithm for Secure DHCPv6 in this document:

Name	Value	RFCs
Non-SigAlg	0x00	this document
RSASSA-PKCS1-v1_5	0x01	this document

Encryption algorithm for Secure DHCPv6. The values in this table are 16-bit unsigned integers. The following initial values are assigned for encryption algorithm for Secure DHCPv6 in this document:

Name	Value	RFCs
Non-EncryAlg	0x00	this document
RSA	0x01	this document

IANA is requested to assign the following new DHCPv6 Status Codes, defined in Section 10.3, in the DHCPv6 Parameters registry maintained in http://www.iana.org/assignments/dhcpv6-parameters:

Code	Name	Reference
TBD9	AuthenticationFail	this document
TBD10	ReplayDetected	this document
TBD11	SignatureFail	this document

13. Acknowledgements

The authors would like to thank Tomek Mrugalski, Bernie Volz, Jianping Wu, Randy Bush, Yiu Lee, Sean Shen, Ralph Droms, Jari Arkko, Sean Turner, Stephen Farrell, Christian Huitema, Stephen Kent, Thomas Huth, David Schumacher, Francis Dupont, Gang Chen, Suresh Krishnan, Fred Templin, Robert Elz, Nico Williams, Erik Kline, Alan DeKok, Bernard Aboba, Sam Hartman, Zilong Liu and other members of the IETF DHC working group for their valuable comments.

This document was produced using the xml2rfc tool [RFC2629].

14. Change log [RFC Editor: Please remove]

draft-ietf-dhc-sedhcpv6-21: Add the reference of draft-ietf-dhc-relay-server-security. Change the SA-ID List as SHA-ID List and delete the HA-id List. The SHA-id List contains the SA-id and HA-id pairs. Add some statements about the Reconfigure message process. Add some specific text on the encryption key tag calculation method; Add more text on security consideration; Changes some mistakes and grammar mistakes

draft-ietf-dhc-sedhcpv6-20: Correct a few grammar mistakes.

draft-ietf-dhc-sedhcpv6-19: In client behavior part, we adds some description about opportunistic security. In this way, in some scenario, authentication is optional. Add the reference of RFC 4034 for the encryption key tag calculation. Delete the part that the relay agent cache server announcements part. Add the assumption that the client’s initial stored increasing number is set to zero. In this way, for the first time increasing number check in the Reply message, the check will always succeed, and then the locally stored number is changed into the contained number in the Reply message. Correct many grammar mistakes.

draft-ietf-dhc-sedhcpv6-18: Add the Algorithm option. The algorithm option contains the EA-id List, SA-id List, HA-id List, and then the certificate and signature options do not contain the algorithm list; Add the Encryption Key Tag option to identify the used public/private key pair; Delete the AlgorithmNotSupported error status code; Delete some description on that secure DHCPv6 exchanges the server selection method; Delete the DecryptionFail error status code; For the case where the client’s certificate is missed, then the server discards the received message. Add the assumption that: For DHCPv6 client, just one certificate is used for the DHCPv6 configuration. Add the statement that: For the first Encrypted-Query message, the server needs to try all the possible private keys and then records the relationship between the public key and the encryption key tag.

draft-ietf-dhc-sedhcpv6-17: Change the format of the certificate option according to the comments from Bernie.

draft-ietf-dhc-sedhcpv6-16: For the algorithm agility part, the provider can offer multiple EA-id, SA-id, HA-id and then receiver choose one from the algorithm set.

draft-ietf-dhc-sedhcpv6-15: Increasing number option only contains the strictly increasing number; Add some description about why encryption is needed in Security Issues of DHCPv6 part;

draft-ietf-dhc-sedhcpv6-14: For the deployment part, Tofu is out of scope and take Opportunistic security into consideration; Increasing number option is changed into 64 bits; Increasing number check is a separate section; IncreasingnumFail error status code is changed into ReplayDetected error status code; Add the section of "caused change to RFC3315";

draft-ietf-dhc-sedhcpv6-13: Change the Timestamp option into Increasing-number option and the corresponding check method; Delete the OCSP stampling part for the certificate check; Add the scenario where the hash and signature algorithms cannot be separated; Add the comparison with RFC7824 and RFC7844; Add the encryption text format and reference of RFC5652. Add the consideration of scenario where multiple DHCPv6 servers share one common DHCPv6 server. Add the statement that Encrypted-Query and Encrypted-Response messages can only contain certain options: Server Identifier option and Encrypted-message option. Add opportunistic security for deployment consideration. Besides authentication+encyrption mode, encryption-only mode is added.

draft-ietf-dhc-sedhcpv6-12: Add the Signature option and timestamp option during server/client authentication process. Add the hash function and signature algorithm. Add the requirement: The Information-request message cannot contain any other options except ORO option. Modify the use of "SHOULD"; Delete the reference of RFC5280 and modify the method of client/server cert verification; Add the relay agent cache function for the quick response when there is no authenticated server. 2016-4-24.

draft-ietf-dhc-sedhcpv6-11: Delete the Signature option, because the encrypted DHCPv6 message and the Information-request message (only contain the Certificate option) don’t need the Signature option for message integrity check; Rewrite the "Applicability" section; Add the encryption algorithm negotiation process; To support the encryption algorithm negotiation, the Certificate option contains the EA-id(encryption algorithm identifier) field; Reserve the Timestamp option to defend against the replay attacks for encrypted DHCPv6

configuration process; Modify the client behavior when there is no authenticated DHCPv6 server; Add the DecryptionFail error code. 2016-3-9.

draft-ietf-dhc-sedhcpv6-10: merge DHCPv6 authentication and DHCPv6 encryption. The public key option is removed, because the device can generate the self-signed certificate if it is pre-configured the public key not the certificate. 2015-12-10.

draft-ietf-dhc-sedhcpv6-09: change some texts about the deployment part.2015-12-10.

draft-ietf-dhc-sedhcpv6-08: clarified what the client and the server should do if it receives a message using unsupported algorithm; refined the error code treatment regarding to AuthenticationFail and TimestampFail; added consideration on how to reduce the DoS attack when using TOFU; other general editorial cleanups. 2015-06-10.

draft-ietf-dhc-sedhcpv6-07: removed the deployment consideration section; instead, described more straightforward use cases with TOFU in the overview section, and clarified how the public keys would be stored at the recipient when TOFU is used. The overview section also clarified the integration of PKI or other similar infrastructure is an open issue. 2015-03-23.

draft-ietf-dhc-sedhcpv6-06: remove the limitation that only clients use PKI- certificates and only servers use public keys. The new text would allow clients use public keys and servers use PKI-certificates. 2015-02-18.

draft-ietf-dhc-sedhcpv6-05: addressed comments from mail list that responsed to the second WGLC. 2014-12-08.

draft-ietf-dhc-sedhcpv6-04: addressed comments from mail list. Making timestamp an independent and optional option. Reduce the serverside authentication to base on only client’s certificate. Reduce the clientside authentication to only Leaf of Faith base on server’s public key. 2014-09-26.

draft-ietf-dhc-sedhcpv6-03: addressed comments from WGLC. Added a new section "Deployment Consideration". Corrected the Public Key Field in the Public Key Option. Added consideration for large DHCPv6 message transmission. Added TimestampFail error code. Refined the retransmission rules on clients. 2014-06-18.

draft-ietf-dhc-sedhcpv6-02: addressed comments (applicability statement, redesign the error codes and their logic) from IETF89 DHC WG meeting and volunteer reviewers. 2014-04-14.

draft-ietf-dhc-sedhcpv6-01: addressed comments from IETF88 DHC WG meeting. Moved Dacheng Zhang from acknowledgement to be co-author. 2014-02-14.

draft-ietf-dhc-sedhcpv6-00: adopted by DHC WG. 2013-11-19.

draft-jiang-dhc-sedhcpv6-02: removed protection between relay agent and server due to complexity, following the comments from Ted Lemon, Bernie Volz. 2013-10-16.

draft-jiang-dhc-sedhcpv6-01: update according to review comments from Ted Lemon, Bernie Volz, Ralph Droms. Separated Public Key/ Certificate option into two options. Refined many detailed processes. 2013-10-08.

draft-jiang-dhc-sedhcpv6-00: original version, this draft is a replacement of draft-ietf-dhc-secure-dhcpv6, which reached IESG and dead because of consideration regarding to CGA. The authors followed the suggestion from IESG making a general public key based mechanism. 2013-06-29.

15. References

15.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

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

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <http://www.rfc-editor.org/info/rfc3971>.

[RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <http://www.rfc-editor.org/info/rfc4034>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <http://www.rfc-editor.org/info/rfc5652>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <http://www.rfc-editor.org/info/rfc5905>.

[RFC6840] Weiler, S., Ed. and D. Blacka, Ed., "Clarifications and Implementation Notes for DNS Security (DNSSEC)", RFC 6840, DOI 10.17487/RFC6840, February 2013, <http://www.rfc-editor.org/info/rfc6840>.

[RFC7283] Cui, Y., Sun, Q., and T. Lemon, "Handling Unknown DHCPv6 Messages", RFC 7283, DOI 10.17487/RFC7283, July 2014, <http://www.rfc-editor.org/info/rfc7283>.

[RFC7296] Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., and T. Kivinen, "Internet Key Exchange Protocol Version 2 (IKEv2)", STD 79, RFC 7296, DOI 10.17487/RFC7296, October 2014, <http://www.rfc-editor.org/info/rfc7296>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7824] Krishnan, S., Mrugalski, T., and S. Jiang, "Privacy Considerations for DHCPv6", RFC 7824, DOI 10.17487/RFC7824, May 2016, <http://www.rfc-editor.org/info/rfc7824>.

[RFC7844] Huitema, C., Mrugalski, T., and S. Krishnan, "Anonymity Profiles for DHCP Clients", RFC 7844, DOI 10.17487/RFC7844, May 2016, <http://www.rfc-editor.org/info/rfc7844>.

15.2. Informative References

[I-D.ietf-dhc-relay-server-security]
Volz, B. and Y. Pal, "Security of Messages Exchanged Between Servers and Relay Agents", draft-ietf-dhc-relay-server-security-03 (work in progress), February 2017.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6273] Kukec, A., Krishnan, S., and S. Jiang, "The Secure Neighbor Discovery (SEND) Hash Threat Analysis", RFC 6273, DOI 10.17487/RFC6273, June 2011, <http://www.rfc-editor.org/info/rfc6273>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <http://www.rfc-editor.org/info/rfc7258>.

[RSA] RSA Laboratories, "RSA Encryption Standard, Version 2.1, PKCS 1", November 2002.

Authors' Addresses

Lishan Li
Tsinghua University
Beijing 100084
P.R.China

Phone: +86-15201441862
Email: lilishan48@gmail.com

Sheng Jiang
Huawei Technologies Co., Ltd
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
CN

Email: jiangsheng@huawei.com

Yong Cui
Tsinghua University
Beijing 100084
P.R.China

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

Tatuya Jinmei
Infoblox Inc.
3111 Coronado Drive
Santa Clara, CA
US

Email: jinmei@wide.ad.jp

Ted Lemon
Nominum, Inc.
2000 Seaport Blvd
Redwood City, CA 94063
USA

Phone: +1-650-381-6000
Email: Ted.Lemon@nominum.com

Dacheng Zhang
Beijing
CN

Email: dacheng.zhang@gmail.com

Internet Engineering Task Force
Internet Draft
Intended status: Standards Track
Expires: September 2017

B. Liu, Ed.
K. Lou
Huawei Technologies
C. Chen
Ericsson

March 7, 2017

Yang Data Model for DHCP Protocol
draft-liu-dhc-dhcp-yang-model-06.txt

Abstract

This document defines a YANG data model for DHCP Server, relay, and client, including configuration and running state.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on September 7, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model that can be used to configure and manage DHCP.

This data model includes configuration data and state data, in which DHCP server, replay, and client nodes are defined.

1.1. Terminology

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].

The following terms are used within this document:

The following terms are defined in [RFC6241] and are not redefined here:

- client
- configuration data
- server

- state data

The following terms are defined in [RFC6020] and are not redefined here:

- augment
- data model
- data node
- presence container

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of Data Model

The goal of this document is to define a data model that provides a common user interface to the DHCP protocol. There is very information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.

2.1. Overview

The overall structure of the model is described as the following.

```
module: ietf-dhcp
   +--rw dhcp
   |  +--rw server
   |  |   ...
   |  |   ...
   |  +--rw relay
```

```
      |  |   ...
      |  |   ...
      |  +--rw client
      |  |   ...
      |  |   ...
      +--ro dhcp-state
         +--ro server
         |  ...
         |  ...
         +--ro relay
         |  ...
         |  ...
         +--ro client
            ...
            ...
```

Furthermore, design three if-feature for deployment of DHCP server/relay/client as below:

```
  feature dhcp-server {
    description "Feature DHCP server";
  }

  feature dhcp-client {
    description "Feature DHCP client";
  }

  feature dhcp-relay {
    description "Feature DHCP relay";
  }
```

2.2. DHCP Server

Server configurations contain lease time, ping packet, IP address pool and option configuration.

The most important part is the IP pool configuration. Specifying IP address section is for dynamic allocation. Configuring the mapping between IP address and MAC address is for manual allocation.

```
module: ietf-dhcp
   +--rw dchp
      +--rw server
         +--rw lease-time?            uint32
         +--rw ping-packet-number?    uint8
         +--rw ping-packet-timeout?   uint16
         +--rw option
         |  +--rw dhcp-server-identifier?   inet:ip-address
```

```
            |  +--rw domain-name?              string
            |  +--rw domain-name-server?       inet:ip-address
            |  +--rw interface-mtu?            uint32
            |  +--rw netbios-name-server?      inet:ip-address
            |  +--rw netbios-node-type?        uint32
            |  +--rw netbios-scope?            string
            +--rw dhcp-ip-pool* [ip-pool-name]
               +--rw ip-pool-name         string
               +--rw interface?           if:interface-ref
               +--rw gateway-ip?          inet:ip-address
               +--rw gateway-mask?        inet:ip-prefix
               +--rw lease-time?          uint32
               +--rw manual-allocation* [mac-address ip-address]
               |  +--rw mac-address    yang:mac-address
               |  +--rw ip-address     inet:ip-address
               +--rw section* [section-index]
               |  +--rw section-index       uint16
               |  +--rw section-start-ip    inet:ipv4-address
               |  +--rw section-end-ip?     inet:ipv4-address
               +--rw option
                  +--rw dhcp-server-identifier?   inet:ip-address
                  +--rw domain-name?              string
                  +--rw domain-name-server?       inet:ip-address
                  +--rw interface-mtu?            uint32
                  +--rw netbios-name-server?      inet:ip-address
                  +--rw netbios-node-type?        uint32
                  +--rw netbios-scope?            string
```

2.3. DHCP Relay

The relay function is configured per interface. Enable/disable relay functionality on a specific interface, and specify the DHCP server.

```
module: ietf-dhcp
   +--rw dchp
      +--rw server
            ...
            ...
      +--rw relay
         +--rw server-group* [server-group-name]
            +--rw server-group-name    string
            +--rw interface?           if:interface-ref
            +--rw gateway-address?     inet:ipv4-address
            +--rw server-address*      inet:ipv4-address
```

2.4. DHCP Client

DHCP client is also managed per interface, including enable/disable client DHCP client function, client id and lease time.

```
module: ietf-dhcp
   +--rw dchp
      +--rw server
            ...
            ...
      +--rw relay
            ...
            ...
      +--rw client
         +--rw interfaces* [interface]
            +--rw interface?   if:interface-ref
            +--rw client-id?   string
            +--rw lease?       uint32
```

2.5. DHCP State

The " dhcp-state" records the package statistic information and host allocated status, including server, relay and client.

```
module: ietf-dhcp
   +--rw dhcp
      +--rw server
            ...
            ...
      +--rw relay
            ...
            ...
      +--rw client
            ...
            ...
   +--ro dhcp-state
      +--ro server {server}?
      |  +--ro packet-statistics
      |  |  +--ro interface?   if:interface-state-ref
      |  |  +--ro receive
      |  |  |  +--ro decline-packet?    uint32
      |  |  |  +--ro discover-packet?   uint32
      |  |  |  +--ro request-packet?    uint32
      |  |  |  +--ro release-packet?    uint32
      |  |  |  +--ro inform-packet?     uint32
```

```
|  |  +--ro send
|  |     +--ro offer-packet?   uint32
|  |     +--ro ack-packet?     uint32
|  |     +--ro nack-packet?    uint32
|  +--ro host
|  |  +--ro interface?               string
|  |  +--ro host-ip?                 string
|  |  +--ro host-hardware-address?   string
|  |  +--ro lease?                   uint32
|  |  +--ro type?                    allocate-type
|  +--ro ip-pool* [ip-pool-name]
|     +--ro ip-pool-name         string
|     +--ro gateway-ip?          inet:ip-address
|     +--ro gateway-mask?        inet:ip-prefix
|     +--ro used-ip-count?       uint32
|     +--ro idle-ip-count?       uint32
|     +--ro conflict-ip-count?   uint32
|     +--ro total-ip-count?      uint32
+--ro relay {relay}?
|  +--ro packet-statistics
|     +--ro interface?   if:interface-state-ref
|     +--ro receive
|     |  +--ro offer-packet?      uint32
|     |  +--ro ack-packet?        uint32
|     |  +--ro nack-packet?       uint32
|     |  +--ro decline-packet?    uint32
|     |  +--ro discover-packet?   uint32
|     |  +--ro request-packet?    uint32
|     |  +--ro release-packet?    uint32
|     |  +--ro inform-packet?     uint32
|     +--ro send
|        +--ro offer-packet?      uint32
|        +--ro ack-packet?        uint32
|        +--ro nack-packet?       uint32
|        +--ro decline-packet?    uint32
|        +--ro discover-packet?   uint32
|        +--ro request-packet?    uint32
|        +--ro release-packet?    uint32
|        +--ro inform-packet?     uint32
+--ro client {client}?
   +--ro packet-statistics
      +--ro interface?   if:interface-state-ref
      +--ro receive
      |  +--ro offer-packet?   uint32
      |  +--ro ack-packet?     uint32
      |  +--ro nack-packet?    uint32
      +--ro send
```

```
                +--ro decline-packet?    uint32
                +--ro discover-packet?   uint32
                +--ro request-packet?    uint32
                +--ro release-packet?    uint32
                +--ro inform-packet?     uint32
```

2.6. DHCP RPC

The " dhcp-rpc" is designed to clean the packet statistics on server/relay/client node.

```
rpcs:
   +---x clean-server-statistics     {dhcp-server}?
   |  +---w input
   |  |  +---w interface?   -> /dhcp-state/server/packet-statistics/interface
   |  |  +---w clean-at?    yang:date-and-time
   |  +--ro output
   |     +--ro clean-finished-at?   yang:date-and-time
   +---x clean-relay-statistics      {dhcp-relay}?
   |  +---w input
   |  |  +---w interface?   -> /dhcp-state/relay/packet-statistics/interface
   |  |  +---w clean-at?    yang:date-and-time
   |  +--ro output
   |     +--ro clean-finished-at?   yang:date-and-time
   +---x clean-client-statistics     {dhcp-client}?
      +---w input
      |  +---w interface?   -> /dhcp-state/client/packet-statistics/interface
      |  +---w clean-at?    yang:date-and-time
      +--ro output
         +--ro clean-finished-at?   yang:date-and-time
```

3. DHCP YANG Module

```
<CODE BEGINS> file "ietf-dhcp@2017-03-02.yang"
module ietf-dhcp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-dhcp";
  prefix "dhcp";

  import ietf-inet-types {
    prefix "inet";
  }
  import ietf-yang-types {
    prefix "yang";
  }
```

```
import ietf-interfaces {
  prefix "if";
}

organization "IETF dhc (Dynamic Host Configuration Protocol)
              Working Group";
contact      "leo.liubing@huawei.com
              loukunkun@huawei.com
              chin.chen@ericsson.com";
description  "The module for implementing DHCP protocol";

revision 2017-03-02 {
  description   "initial draft revision";
  reference     "rfc2131 rfc6020";
}

/*--------------------*/
/*      Features      */
/*--------------------*/
feature dhcp-server {
  description "Feature DHCP server";
}

feature dhcp-client {
  description "Feature DHCP client";
}

feature dhcp-relay {
  description "Feature DHCP relay";
}

/*--------------------*/
/*  Types Defination  */
/*--------------------*/
typedef allocate-type {
  type enumeration {
    enum automatic {
    description
      "DHCP assigns a permanent IP address to a client";
    }
    enum dynamic {
    description
      "DHCP assigns an IP address to a client
       for a limited period of time";
    }
    enum manual {
    description
```

```
            "a client's IP address is assigned by the
                       network administrator, and DHCP is used
             simply to convey the assigned address to the client";
          }
        }
        description "Mechanisms for IP address allocation";
      }

      /*--------------------*/
      /*      Groupings     */
      /*--------------------*/
      grouping server-packet {
        description "The packets are sent from server ";

        leaf offer-packet {
          type uint32;
          config "false";
          description "Total number of DHCPOFFER packets";
        }
        leaf ack-packet {
          type uint32;
          config "false";
          description "Total number of DHCPACK packets";
        }
        leaf nack-packet {
          type uint32;
          config "false";
          description "Total number of DHCPNAK packets";
        }
      }

      grouping client-packet {
        description "The packets are sent from client ";

        leaf decline-packet {
          type uint32;
          config "false";
          description "Total number of DHCPDECLINE packets";
        }
        leaf discover-packet {
          type uint32;
          config "false";
          description "Total number of DHCPDISCOVER packets";
        }
        leaf request-packet {
          type uint32;
          config "false";
```

```
        description "Total number of DHCPREQUEST packets";
      }
      leaf release-packet {
        type uint32;
        config "false";
        description "Total number of DHCPRELEASE packets";
      }
      leaf inform-packet {
        type uint32;
        config "false";
        description "Total number of DHCPINFORM packets";
      }
    }

    grouping sum-packet {
      description "All of commnicated packets between server and client";

      uses server-packet;

      uses client-packet;
    }

    grouping dhcp-option {
      description "Configuration option";

      leaf dhcp-server-identifier {
        type  inet:ip-address;
        description "DHCP server identifier";
      }
      leaf domain-name {
        type  string;
        description "Name of the domain";
      }
      leaf domain-name-server {
        type  inet:ip-address;
        description "IPv4 address of the domain";
      }
      leaf interface-mtu {
        type  uint32 {
          range "0..65535";
        }
        description "Minimum Transmission Unit (MTU) of the interface";
      }
      leaf netbios-name-server {
        type  inet:ip-address;
        description "NETBIOS name server";
      }
```

```
      leaf netbios-node-type {
        type  uint32 {
          range "0..65535";
        }
        description "NETBIOS node type";
      }
      leaf netbios-scope {
        type  string;
        description "NETBIOS scope";
      }
    }

    /*--------------------*/
    /* Configuration Data */
    /*--------------------*/
    container dhcp {
      description
        "DHCP configuration";
      container server {
        if-feature dhcp-server;
        description
        "DHCP server configuration";
        leaf lease-time {
          type uint32{
             range "180..31536000";
           }
           description
             "Default network address lease time assigned to DHCP clients";
        }
        leaf ping-packet-number{
          type uint8 {
             range "0..10";
           }
           default "0";
           description "Number of ping packets";
        }
        leaf ping-packet-timeout {
          type uint16 {
            range "0..10000";
          }
          default "500";
          description "Timeout of ping packet";
        }
        container option {
          description "Configuration option";
          uses dhcp-option;
        }
```

```
          list dhcp-ip-pool {
            key "ip-pool-name";
            description   "Global IP pool configuration";

            leaf ip-pool-name {
              type string {
                length "1..64";
              }
              description "Name of the IP pool";
            }
            leaf interface {
              type if:interface-ref;
              description
                "Name of the interface";
            }
            leaf gateway-ip {
              type inet:ip-address;
              description "IPv4 address of the gateway";
            }
            leaf gateway-mask {
              type inet:ip-prefix;
              description "Network submask of the gateway";
            }
            leaf lease-time {
              type uint32 {
                range "180..31536000";
              }
              description
                "Default network address lease time assigned to DHCP clients";
            }
            list manual-allocation {
              key "mac-address ip-address";
              description "Mapping from MAC address to IP address";

              leaf mac-address {
                type yang:mac-address;
                description "MAC address of the host";
              }
              leaf ip-address {
                type inet:ip-address;
                description "IPv4 address of the host";
              }
            }
            list section {
              key "section-index";
              description "IPv4 address for the range";
              leaf section-index {
```

```
            type uint16 {
                    range "0..255";
            }
            description "Index of IPv4 address range";
          }
          leaf section-start-ip {
            type inet:ipv4-address;
            mandatory "true";
            description "Starting IPv4 Address of a section";
          }
          leaf section-end-ip {
            type inet:ipv4-address;
            description "Last IPv4 Address of a section";
          }
        }
        container option {
          description "Configuration option";
          uses dhcp-option;
        }
      }
    }
    container relay {
      if-feature dhcp-relay;
      description "DHCP relay agent configuration";

      list server-group {
        key "server-group-name";
        description
          "DHCP server group configuration that DHCP relays to";
        leaf server-group-name {
          type string;
          description "Name of a DHCP server group";
        }
        leaf interface {
          type if:interface-ref;
          description "Name of the interface";
        }
        leaf gateway-address {
          type inet:ipv4-address;
          description "IPv4 address of the gateway";
        }
        leaf-list server-address {
          type inet:ipv4-address;
          description "IPv4 address of the server";
        }
      }
    }
```

```
       container client {
         if-feature dhcp-client;
         description "DHCP client configuration";

         list interfaces {
           key "interface";
           description "Interface configuration";

           leaf interface {
             type if:interface-ref;
             description "Name of the interface";
           }
           leaf client-id {
             type string;
             description "DHCP client identifier";
           }
           leaf lease {
             type uint32 {
               range "1..4294967295";
             }
             description "Default network address lease time assigned to DHCP cl
ients";
           }
         }
       }
     }

     /*--------------------------*/
     /*  Operational State Data  */
     /*--------------------------*/
     container dhcp-state {
       config "false";
       description "DHCP state data";

       container server {
         if-feature dhcp-server;
         description "DHCP server state data";

         container packet-statistics {
           description "Packet statistics";

           leaf interface {
             type if:interface-state-ref;
             description "Name of the interface";
           }

           container receive {
             description "Number of  received packets";
```

```
            uses client-packet;
          }
          container send {
            description "Number of sent packets";

            uses server-packet;
          }
        }
        container host {
          description "Host status information";
          leaf interface {
            type string;
            config "false";
            description "Name of the interface";
          }
          leaf host-ip {
            type string;
            config "false";
            description "IPv4 address of the host";
          }
          leaf host-hardware-address {
            type string;
            config "false";
            description "MAC address of the host";
          }
          leaf lease {
            type uint32;
            config "false";
            description "Default network address lease
                         time assigned to DHCP clients";
          }
          leaf type {
            type allocate-type;
            config "false";
            description "Mechanisms for IP address allocation";
          }
        }
        list ip-pool {
          key "ip-pool-name";
          description   "Global IP pool configuration";

          leaf ip-pool-name {
            type string {
              length "1..64";
            }
            description "Name of an IP pool";
```

```
          }
          leaf gateway-ip {
            type inet:ip-address;
            description "IPv4 address of the gateway";
          }
          leaf gateway-mask {
            type inet:ip-prefix;
            description "Network submask of the gateway";
          }
          leaf used-ip-count {
            type uint32;
            config "false";
            description "Total number of used IPv4 addresses";
          }
          leaf idle-ip-count  {
            type uint32;
            config "false";
            description "Total number of idle IPv4 addresses";
          }
          leaf conflict-ip-count {
            type uint32;
            config "false";
            description "Total number of conflict IPv4 addresses";
          }
          leaf total-ip-count  {
            type uint32;
            config "false";
            description "Total number of IPv4 addresses";
          }
        }
      }
      container relay {
        if-feature dhcp-relay;
        description "DHCP reply agent state data";

        container packet-statistics {
          description "Packet statistics";

          leaf interface {
            type if:interface-state-ref;
            description "Name of the interface";
          }

          container receive {
            description "Number of  received packets";

            uses sum-packet;
```

```
          }
          container send {
            description
              "Number of sent packets";

            uses sum-packet;
          }
        }
      }
      container client {
        if-feature dhcp-client;
        description "DHCP client state data";

        container packet-statistics {
          description "Packet statistics";

          leaf interface {
            type if:interface-state-ref;
            description "Name of the interface";
          }

          container receive {
            description "Number of  received packets";

            uses server-packet;
          }
          container send {
            description "Number of sent packets";

            uses client-packet;
          }
        }
      }
    }

    /*---------------------------*/
    /*  Define RPC for action   */
    /*---------------------------*/
    rpc clean-server-statistics {
      if-feature dhcp-server;
      description "Clean server packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:server/"+
              "dhcp:packet-statistics/dhcp:interface";
          }
```

```
          description "Name of the interface";
        }
        leaf clean-at {
          type yang:date-and-time;
          description "The start time to clean packet statistics";
        }
      }
      output {
        leaf clean-finished-at {
          type yang:date-and-time;
          description "The finish time to clean packet statistics";
        }
      }
    }

    rpc clean-relay-statistics {
      if-feature dhcp-relay;
      description "Clean relay packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:relay/"+
              "dhcp:packet-statistics/dhcp:interface";
          }
          description "Name of the interface";
        }
        leaf clean-at {
          type yang:date-and-time;
          description "The start time to clean packet statistics";
        }

      }
      output {
        leaf clean-finished-at {
          type yang:date-and-time;
          description "The finish time to clean packet statistics";
        }
      }
    }

    rpc clean-client-statistics {
      if-feature dhcp-client;
      description "Clean client packet statistics";
      input {
        leaf interface {
          type leafref {
              path "/dhcp:dhcp-state/dhcp:client/"+
```

```
                "dhcp:packet-statistics/dhcp:interface";
            }
            description "Name of the interface";
          }
          leaf clean-at {
            type yang:date-and-time;
            description "The start time to clean packet statistics";
          }

        }
        output {
          leaf clean-finished-at {
            type yang:date-and-time;
            description "The finish time to clean packet statistics";
          }
        }
      }
    }
    <CODE ENDS>
```

4. Security Considerations

The data model defined does not create any security implications.

5. Contributors

The following people all contributed significantly to the initial

YANG model:

- Gang Yan (Huawei)

- Hongying Sheng (Ericsson)

6. IANA Considerations

This draft does not request any IANA action.

7. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <http://www.rfc-editor.org/info/rfc2629>.

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

[RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, DOI 10.17487/RFC6021, October 2010, <http://www.rfc-editor.org/info/rfc6021>.

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

Authors' Addresses

Bing Liu
Huawei Technologies
Q14, Huawei Campus, No.156 Beiqing Road
Hai-Dian District, Beijing, 100095
P.R. China

Email: leo.liubing@huawei.com

Kunkun Lou
Huawei Technologies
Huawei Nanjing R&D Center
101 Software Avenue, Yuhua District, Nanjing, Jiangsu, 210012
P.R. China

Email: loukunkun@huawei.com

Chin Chen
Ericsson (China) Communications Company Ltd.
Ericsson Tower, No. 5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email: chin.chen@ericsson.com

Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

G. Ren
L. He
Y. Liu
Tsinghua University
March 13, 2017

Multi-requirement Extensions for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)
draft-ren-dhc-mredhcpv6-00

Abstract

This memo provides multi-requirement extensions for DHCPv6, which allow hosts to generate or fetch addresses according to the user or network requirements, and DHCP servers to centrally manage all types of addresses including SLAAC-configured addresses, DHCPv6-configured addresses, and manual-configured addresses. Moreover, a general extension for address generation is designed to allow multiple types of requirements to be introduced into the DHCPv6 exchanges.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Table of Contents

1. Introduction

There are two address auto-configuration methods in IPv6: Stateless Address Autoconfiguration (SLAAC) [RFC4862], and the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315]. Several address generation mechanisms have been proposed, including IEEE EUI-64 [RFC2464], CGAs [RFC3972], Temporary [RFC4941], and Stable, privacy [RFC7217]. The many types of IPv6 address generation and configuration methods available have brought about flexibility and diversity.

However, the current IPv6 address assignment and management are still confronted with certain problems, including a mixed operation problem of multiple IPv6 address generation mechanisms, a synchronization problem with a change in IPv6 addresses, an efficiency problem of processing large-scale concurrent IPv6 address requests, and a general model problem in introducing external services into the address assignment process.

Faced with various network requirements, various entities that prefer to remain up to date on all types of addresses, and various extensions for external services, it is important to balance the flexibility of address generation and configuration, user privacy, and network manageability. To solve the four problems above, the multi-requirement extensions for DHCPv6 are proposed, which can be achieved by extending DHCPv6 under the premise of changing the current protocols as little as possible.

This memo provides multi-requirement extensions for DHCPv6, which allow hosts to generate addresses through SLAAC or fetch addresses assigned from DHCPv6 servers according to the user or network requirements. At the same time, the extensions allow DHCP servers to centrally manage all kinds of addresses including SLAAC-configured addresses, DHCPv6-configured addresses, and manual-configured addresses. Moreover, a general extension for address generation is designed to allow multiple types of requirements to be introduced into the DHCPv6 exchanges.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this memo are to be interpreted as described in [RFC2119].

Familiarity with DHCPv6 and its terminology, as defined in [RFC3315], is assumed.

Other terminology:

Requirements	A functional need that a particular design or process of the address generation and assignment must be able to perform.
External Services	Services that are introduced into the DHCP exchanges according to specific requirements, such as traceback, transition, and mobility.
External Service Client	An entity requesting a specific external service.
External service Server	An entity providing a specific external service to external service clients.

3. Problem Statement

3.1. Address Configuration Methods

SLAAC and DHCPv6 are two auto-configuration mechanisms in IPv6 [RFC2460]. SLAAC can configure hosts with one or more addresses composed of a network prefix advertised by a local router, and an Interface Identifier (IID) that typically embeds a hardware address (e.g., an IEEE LAN MAC address) [RFC4291]. DHCPv6 can provide a device with addresses assigned by a DHCPv6 server and other configuration information carried in the options.

3.2. Address Generation Mechanisms

Several IID generation mechanisms have been proposed for standardization, all of which have their own specific requirements. IEEE 64-bit Extended EUI-64 [RFC2464] generates IIDs based on IEEE 802 48-bit Media Access Control (MAC) addresses quickly and economically. Cryptographically Generated Addresses (CGAs) [RFC3972] are another method for generating an IID, which binds a hash of the host’s public key to an IPv6 address in the SEcure Neighbor Discovery (SEND) protocol [RFC3971]. The owners of CGAs can sign messages using the corresponding private keys to protect their messages. Temporary addresses defined in [RFC3041] (later made obsolete by [RFC4941]) are randomly generated for outgoing connections to protect the host’s privacy, and are changed daily. However, temporary addresses make it difficult to conduct network management (e.g.,

increase the complexity of event logging and access controls). Therefore, [RFC7217] specifies an algorithm that generates a unique random stable, semantically opaque IID per IPv6 link for each network without sacrificing the security and privacy of users. The DHCPv6 variant of this method is specified in [RFC7943].

3.3. Determinants of IPv6 Address Allocations

3.3.1. Routers

The ICMPv6-based [RFC4443] Router Advertisement (RA) message specified in Neighbor Discovery (ND) [RFC4861] contains M and A flags that allow a host to generate or fetch different types of addresses (SLAAC addresses and DHCPv6 addresses) when they are set. At the same time, several routers may exist in the same network, all with different flags and prefix settings.

3.3.2. DHCPv6 Servers

When the M flag is set, it indicates that addresses are available through DHCPv6 service. In fact, there may be several DHCPv6 servers in a network that can assign addresses to DHCPv6 clients. Each DHCPv6 server can assign addresses of different types, non-temporary and temporary, and different policies, including iterative, identifier-based, hash, and random [RFC7824].

3.3.3. Hosts

A host can configure its addresses in the following ways:

Manual Administrators can configure static addresses for the host.

SLAAC When A flags in the prefix information options (PIOs) of an RA message are set, a host can utilize the prefixes in the PIOs of RA messages to generate addresses automatically. Many different address generation mechanisms can be utilized, including IEEE EUI-64 [RFC2464], CGA[RFC3972], Temporary [RFC4941], and Stable, privacy[RFC7217].

3.4. Address-Related Network Entities

After address assignments, many other network service entities record and maintain data entries related to the addresses.

Switches Switches will create entries for the addresses in the forwarding table.

DHCPv6 servers	DHCPv6 servers will record and maintain the address leases for the clients.
Auditing server	An auditing server will record the detailed traffic usage of the addresses.
Gateway	A gateway will use the authenticated address list to control the host's access to the Internet.
Other External servers	Other external service servers may need to maintain the address-related information.

3.5. Current Problems

The current IPv6 address assignment and management are still confronted with certain problems, including mixed operation problem, synchronization problem, efficiency problem, and general model problem.

3.5.1. Mixed Operation Problem

The first problem is a mixed operation problem of multiple IPv6 address generation mechanisms. Currently, even one host interface can have several addresses generated from different address generation mechanisms. Moreover, hosts in the same network can use different address generation mechanisms for SLAAC to obtain addresses and/or fetch addresses assigned from DHCPv6 servers. Requirements exist for networks to uniformly configure their address generation mechanism. For SLAAC addresses, difficulties arise when conducting address management and some other network services (e.g., authentication), because most operating systems leverage temporary addresses, which vary over time (e.g., after one day). At the same time, persistent connections will be cut off when the addresses vary. To summarize, the addresses of the hosts can be generated according to not only the user requirements but also to the network requirements.

3.5.2. Synchronization Problem

The second problem is a synchronization problem with a change in IPv6 addresses. Many types of network function entities related to addresses exist, as mentioned in Section 3.4. Once a host updates its addresses, or if the address that the host is currently using is re-assigned to another user for a particular reason (e.g., without renewing the lease), the corresponding function entities should also update their corresponding stored entries. [RFC7653] solves a part of this problem by allowing other network entities to keep up with

the DHCPv6 leases. However, the part of the problem caused by SLAAC and manual configurations remains unresolved.

3.5.3. Efficiency Problem

The third problem is an efficiency problem when processing large-scale concurrent IPv6 address requests. When large-scale concurrent IPv6 address requests exist, the routers will use more resources to forward the multicast messages when the hosts conduct duplicate address detection (DAD) [RFC4862]. However, when central management is used for all types of addresses, this address management entity can detect duplicate addresses for the hosts. It is simple to choose between the concurrent processing mechanism of server clustering techniques and the current DAD processing mode, which puts significant pressure on the routers when large-scale concurrent address requests exist.

3.5.4. General Model Problem

The fourth problem is a general model problem in introducing external services into the address assignment process. On the one hand, IP addresses are not only locators they are also identifiers. As identifiers, IP addresses can be mapped to other requirement spaces to support multiple functions, such as traceback, transition, and mobility. On the other hand, some interoperations between DHCP entities with external service entities are designed to provide precise and fine-grained services. For example, IETF defines the interoperations between DHCPv6 relays and radius servers [RFC7037] to provide authorization and identification information between the DHCPv6 relay agent and DHCPv6 server. In short, there are no general uniform protocol extensions or models for introducing external services into the address assignment process.

4. Design Goals

To solve the above problems, the solution should achieve the following goals:

Goal 1 Addresses in a network should be generated according to the user or network requirements. In fact, DHCPv6 servers already assign addresses according to these requirements. DHCPv6 servers can assign temporary or non-temporary addresses to DHCPv6 clients. DHCPv6 also provides several address allocation policies according to the administrative requirements and settings, including iterative, identifier-based, hash and random [RFC7824].

Goal 2 All types of addresses in a network should be within the central management, including SLAAC-configured addresses, DHCPv6-configured addresses, and manual-configured addresses. Because a DHCPv6 server manages a pool of IPv6 addresses and information regarding client configuration parameters, it will be a good option for the DHCPv6 server to manage other types of addresses when necessary.

Goal 3 General uniform protocol extensions and models for introducing external services into the process of address assignment should be built.

5. General Structures

5.1. General Address Generation Type

According to Section 3.2, the general address generation types are summarized below.

Type	Method	Related RFC
1	IEEE EUI-64	RFC 2464
2	CGAs	RFC3972
3	Temporary	RFC4941
4	Stable, privacy	RFC7217/RFC7943

Table 1: General address generation types

5.2. Uniform Address Storage Structure

Because DHCPv6 servers will store all kinds of address assignments, it is necessary to design a uniform address assignment storage structure. Several key elements have been selected to construct the core of the address assignment storage structure.

address IPv6 address.

duid DHCP unique identifier, see Section 9 of [RFC3315].

iaid Identity association, see Section 10 of [RFC3315].

valid_lifetime	Length of the lease.
expire	Expiration time of the lease.
pref_lifetime	Preferred lifetime.
hwaddr	Hardware/MAC address.

address	duid	iaid	valid_lifetime	expire	pref_lifetime	hwaddr

For SLAAC address assignments and manual address configurations, some information may be absent, including duid and iaid. Other information can also be included in the uniform address storage structure, such as the subnet identification, hostname, and lease type.

6. Solutions

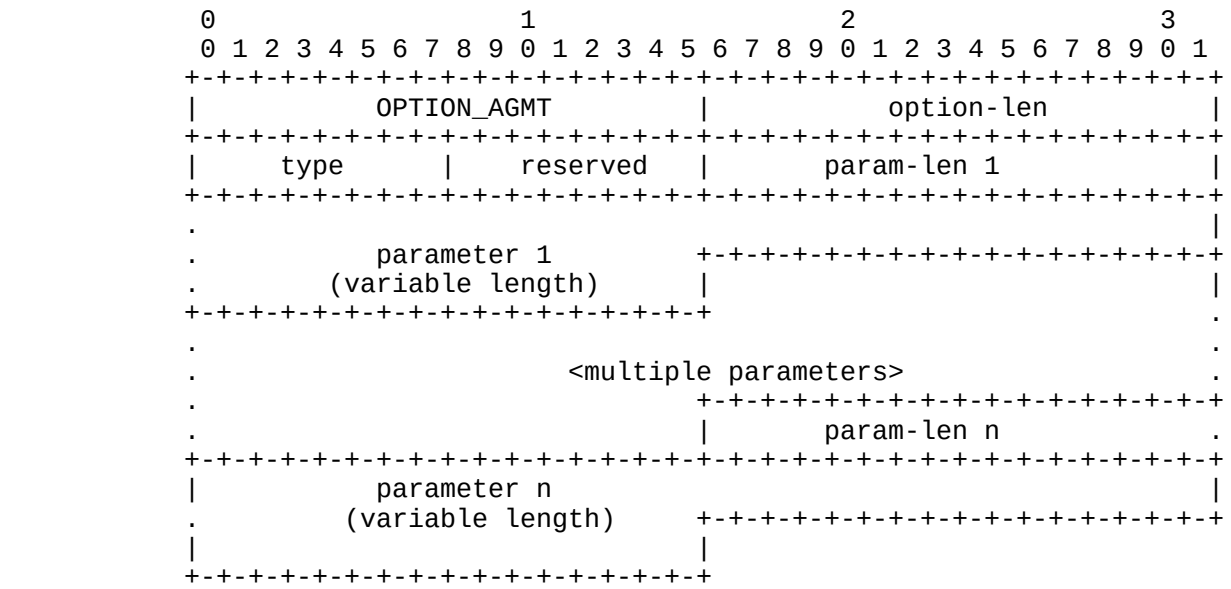
For Goal 1, mechanisms that allow SLAAC-configured addresses and manual-configured addresses to be sent to the DHCPv6 server can be provided. For Goal 2, extensions for both SLAAC and DHCPv6 should be provided. More specifically, extensions of PIO are provided for SLAAC, and new options have been designed for DHCPv6. For Goal 3, a general address generation extension for DHCPv6 is presented herein.

6.1. Sub-solution for DHCPv6

6.1.1. Extension of DHCPv6 Options

6.1.1.1. Address Generation Mechanism Type Option

A server sends this option to inform the client of the address generation mechanism used in the administrative domain. The format of the Address Generation Mechanism Type (AGMT) option is as follows:



Format description:

option-code OPTION_AGMT(TBA1).

option-len 2 + Length of following multiple parameters in octets.

type IID generation mechanism type that the server selects. A value of zero is assigned as the default value.

0 Any IID generation mechanism type.

1 IEEE EUI-64.

2 CGAs.

3 Temporary addresses.

4 Stable, semantically opaque IIDs.

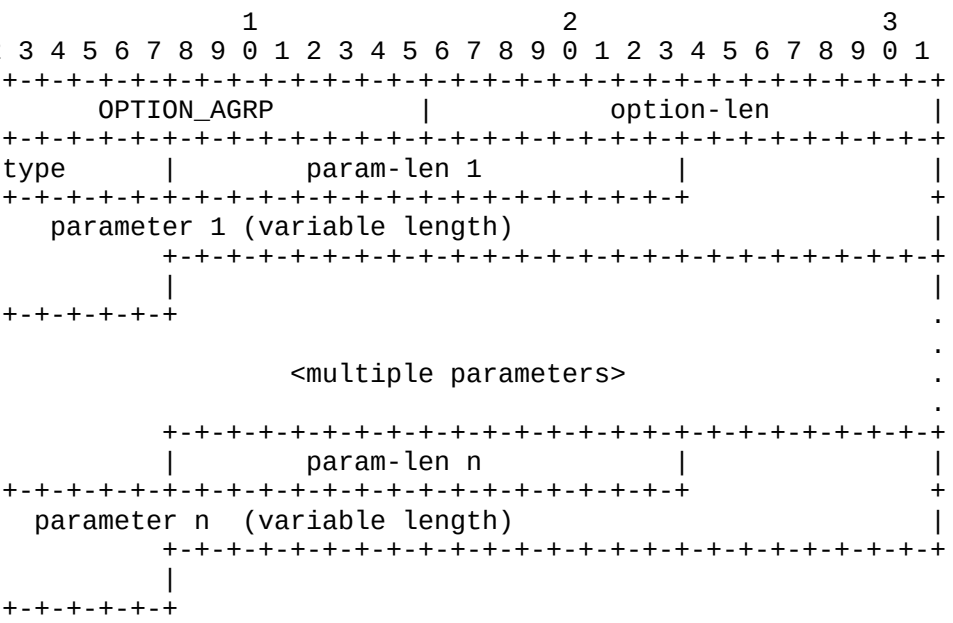
reserved Reserved field for future extensions. The server MUST set this value to zero, and the client MUST ignore its content.

param-len 1...n This is a 16-bit integer that specifies the length of the following parameters in octets (not including the parameter-length field).

parameter 1...n These UTF-8 strings are parameters needed for servers to inform the clients according to the selected address generation mechanisms. The strings are not NUL-terminated.

6.1.1.2. Address Generation Requiring Parameters Option

The client sends this option to inform the server of the parameters of the corresponding address generation mechanism. The format of the Address Generation Requiring Parameters (AGRP) option is as follows:



Format description:

option-code OPTION_AGRP(TBA2).

option-len 1 + Length of following multiple parameters in octets.

type IID generation mechanism type selected by the server.

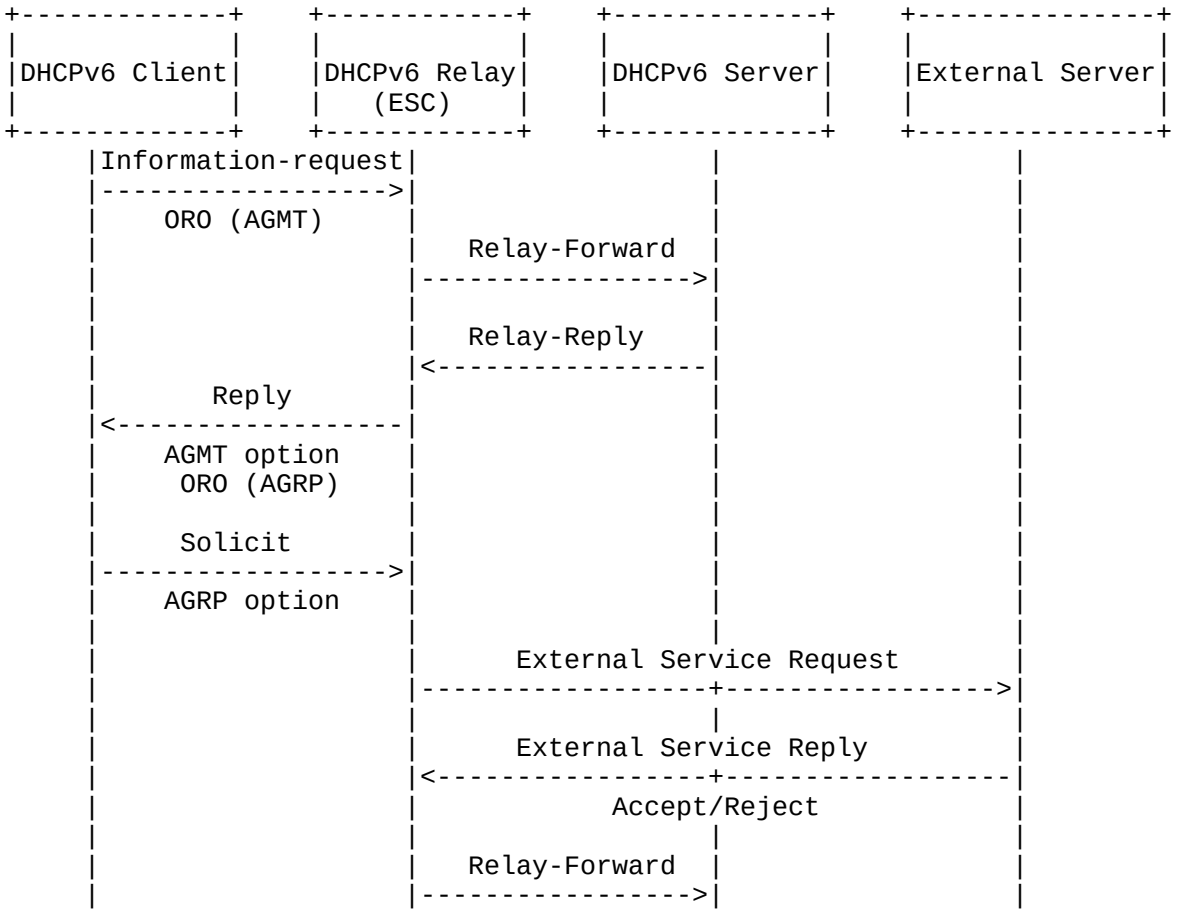
param-len 1...n This is a 16-bit integer that specifies the length of the following parameter in octets (not including the parameter-length field).

parameter 1...n These UTF-8 strings are parameters needed for the clients to inform the servers of according to the selected address generation mechanism. The strings are not NUL-terminated.

6.1.2. Extension of DHCPv6 Exchange Process

6.1.2.1. Overview

The part of the intent of this memo is to dynamically configure the address generation mechanism. The following figure illustrates the new DHCPv6 exchange process. Briefly, a client requests the address generation mechanism from servers. The servers tell the client which type of address generation mechanism to use. Some parameters can also be sent to the servers to generate new addresses when the clients start to request addresses.



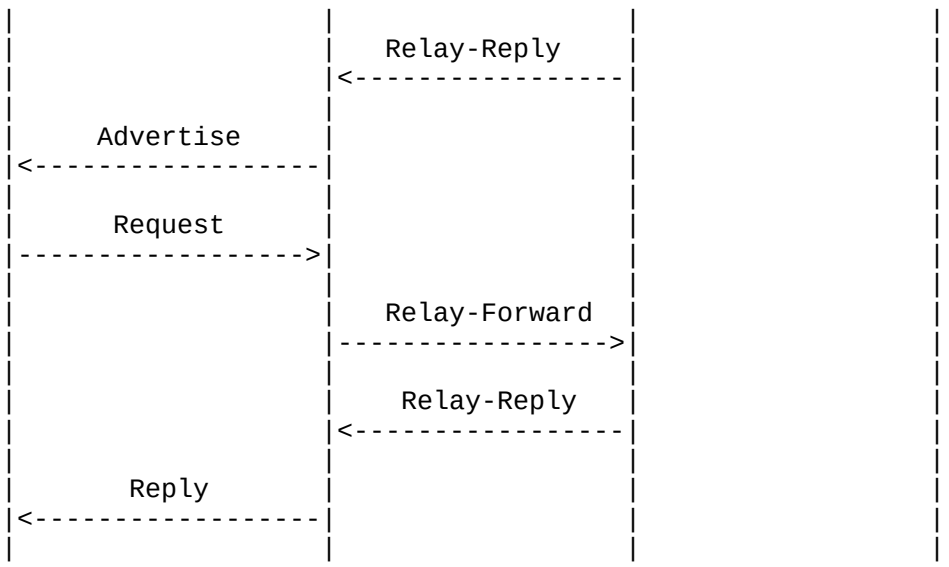


Figure 1: Extension of DHCPv6 Exchange Process

6.1.2.2. Detailed Exchanges

The detailed exchanges of the extensions specified in this memo are as follows:

1 A client requests the AGMT option in the Option Request Option (ORO), which is carried in an Information-request message.

2 The relay agent forwards the Information-request message to the servers.

3 The servers tell the client which type of address generation mechanism to use through the AGMT option within the Reply message. If a server requires the client to provide parameters to generate the addresses, it requests the AGRP option in the ORO.

4 The relay agent forwards the Reply message to the client.

5 If the address generation mechanism selected by the server does not require the client to send other parameters, the client sends a normal Solicit message. Otherwise, the client sends a Solicit message with an AGRP option.

6 When the relay agent receives a Solicit message, it checks whether the selected method requires communication with external servers. If not, it forwards the message to the server. Otherwise, it communicates with the external server to finish the related task (e.g., authentication or authority) as an external service client (ESC) (see Section 6.1.3). Next, it forwards the Solicit message to the server if the communication process succeeds. Otherwise, it drops the message.

7 If there is an AGRP option in the Solicit message, the server uses the parameters in the AGRP option to generate an address and sends an Advertise message with the address to the client. Otherwise, the server handles the message based on [RFC3315].

8 The remaining steps are the same as with the original DHCPv6 process.

6.1.3. Extension of External Service Result Fetching Process

The figure below shows the communication process between a DHCPv6 relay, or server, and an ESC, which only includes two messages: an External Service Request message and an External Service Reply message. Notice that the ESC can be located on the same device with the DHCPv6 relay agent or DHCPv6 server.

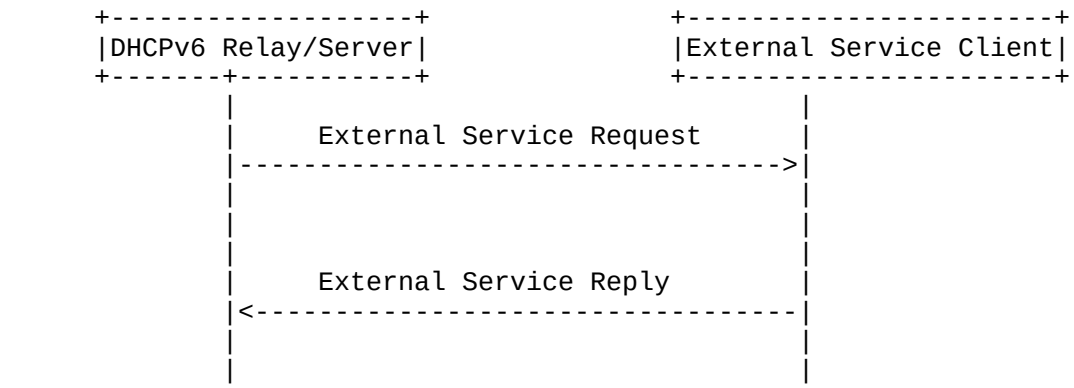


Figure 2: Extension of External Service Result Fetching Process

The format of the External Service Message is as follows:

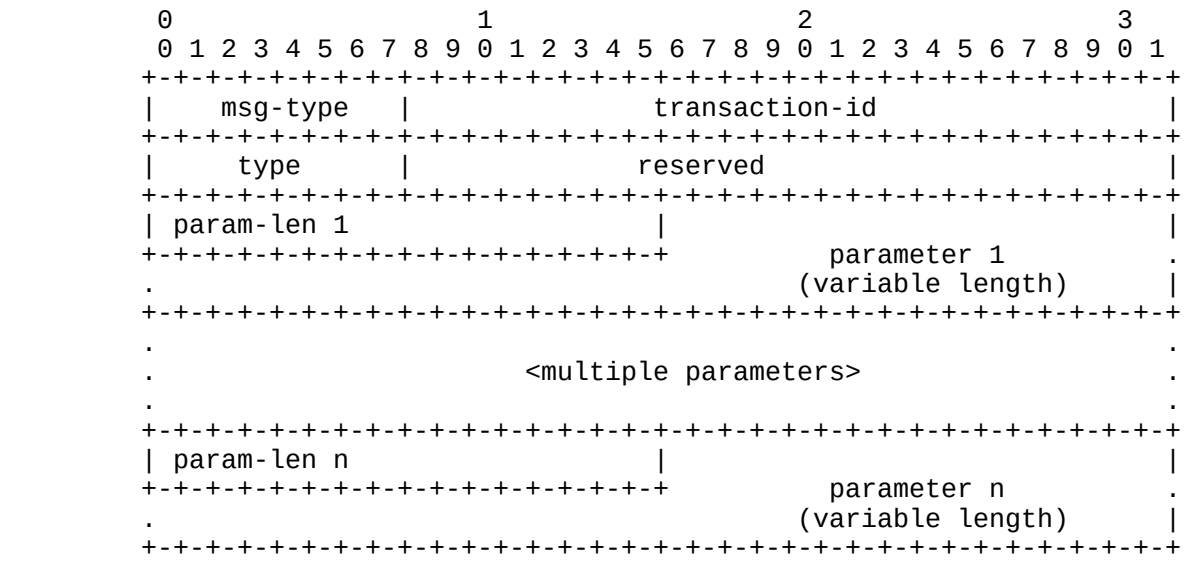


Figure 3: External Service Message Format

Format description:

msg-type The message type, including EXTERNAL_SERVICE_REQUEST(TBA3) and EXTERNAL_SERVICE_REPLY(TBA4).

transaction-id The transaction id copied from a Solicit message to identify this message exchange.

type External service type.

reserved Reserved field for future extensions. The server MUST set this value to zero, and the client/server MUST ignore its content.

param-len 1...n This is a 16-bit integer that specifies the length of the following parameter in octets (not including the parameter-length field).

parameter 1...n These UTF-8 strings are parameters required for external services. The strings are not NUL-terminated.

6.1.3.1. External Service Request Message

This message is used by the DHCPv6 relay or server to request an external service at the external server. The DHCPv6 relay or server sends this message to the ESC, which extracts the parameters in the message to start an external service communication process. For example, when the DHCPv6 process uses a radius server to authenticate or authorize a client [RFC7037], the message can be used to send the relevant parameters to the radius client.

When the DHCP relay or server creates such a message, it sets the msg-type to EXTERNAL_SERVICE_REQUEST and the type to a specific external service type, copies the transaction-id from the message triggering the external service, and provides the specific parameters required by the external service to the external service client.

6.1.3.2. External Service Reply Message

This message is used by the ESC to reply to the DHCPv6 relay or server with the acceptance or rejection result of the external service.

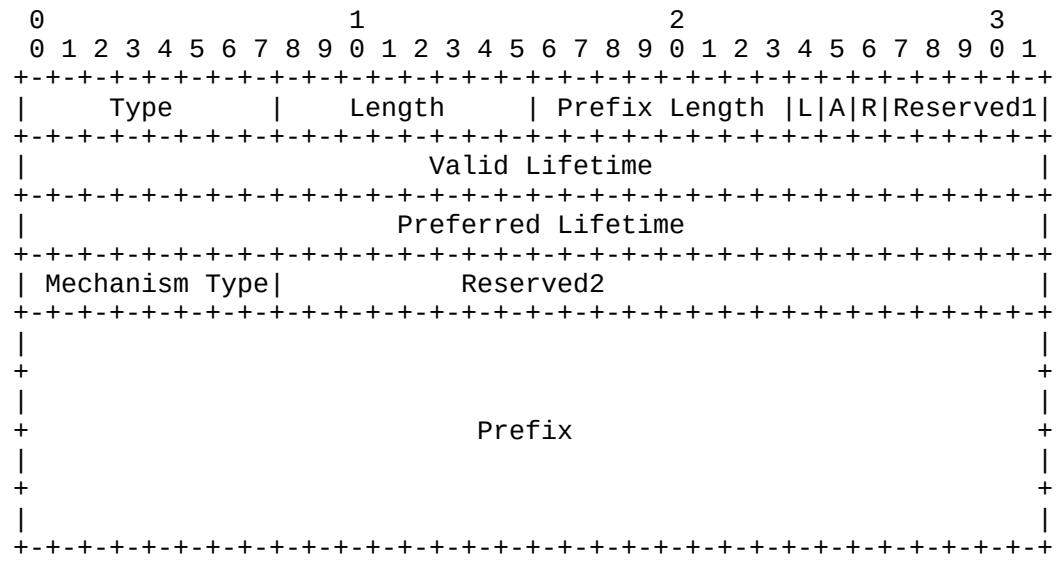
When the external service client creates the message, it sets the msg-type to EXTERNAL_SERVICE_REPLY, copies the transaction-id and type from the External Service Request message, and provides the specific result parameters to the DHCPv6 relay or server.

6.2. Sub-solution for SLAAC

6.2.1. Extension of RA Options

6.2.1.1. Modified Prefix Information Option Format

To support multiple requirements in the address generation for SLAAC, Neighbor Discovery [RFC4861] can be extended to allow a router to advertise the default address generation mechanism for each prefix through the addition of one octet in the format of a PIO for use in the Router Advertisement messages. The format of the PIO is as follows:



This format represents the following changes over that originally specified for Neighbor Discovery [RFC4861]:

Scheme Type 1-octet address generation mechanism type flag. When the A flag is set, it indicates that the PIO recommends the hosts to use the address generation mechanism specified in the Mechanism Type and the prefix specified in Prefix to generate the addresses.

0 Any IID generation mechanism type.

1 IEEE EUI-64.

2 CGAs.

3 Temporary.

4 Stable, privacy.

Reserved2 Reduced from a 4-octet field to a 3-octet field to account for the addition of the above octet.

6.2.2. Extension of Hosts

The hosts should implement the standardized address generation mechanisms mentioned in Section 5.1.

When a host receives RA messages containing a modified PIO, it handles the message based on [RFC4861]. It uses the recommended mechanism type in the PIO to generate the addresses. Note that multiple PIOs may recommend different address generation mechanisms.

6.2.3. Central Management of SLAAC-configured Address

After finishing the address generation, a host should inform the DHCPv6 server of its SLAAC-configured addresses or manual-configured addresses to help the DHCPv6 server manage all types of addresses in the network. There are several schemes to consider:

Scheme 1: Create two new messages similar to Request and Reply messages of DHCPv6 to inform the DHCPv6 server of the SLAAC-configured or manual-configured addresses.

Scheme 2: Use and modify a current mechanism to inform the DHCPv6 server of the addresses. For example, because every SLAAC-configured address performs a DAD, a Neighbor Solicit message can be modified to support this function.

7. Security Considerations

The known security vulnerabilities of the DHCPv6 protocol may apply to its options. Security issues related with DHCPv6 are described in Section 23 of [RFC3315].

Network administrators should be aware that certain external service messages are encrypted, and that DHCPv6 messages are always unencrypted. It is possible for some external service attributes to contain sensitive or confidential information. Network administrators are strongly advised to prevent such information from being included in DHCPv6 messages.

[secure_dhcpv6] provides a new method for protecting end-to-end communication using public key cryptography.

8. IANA Considerations

This memo defines two new DHCPv6 [RFC3315] messages types. The IANA is requested to assign values for the two messages types in the registry maintained in http://www.iana.org/assignments/dhcpv6-parameters: The two new messages type are:

The EXTERNAL_SERVICE_REQUEST(TBA3), described in Section Figure 3.

The EXTERNAL_SERVICE_REPLY(TBA4), described in Section Figure 3.

This memo defines two new DHCPv6 [RFC3315] options. The IANA is requested to assign values for the two options from the DHCPv6 Options Codes table of the DHCPv6 Parameters registry maintained in http://www.iana.org/assignments/dhcpv6-parameters. The two options are:

The Address Generation Mechanism Type option (TBA1), described in Section Section 6.1.1.1.

The Address Generation Requiring Parameters option(TBA2), described in Section Section 6.1.1.2.

IID generation mechanism for multi-requirement extension for DHCPv6. The values in this table are 8-bit unsigned integers. The following initial values are assigned for IID generation mechanism for multi-requirement extension for DHCPv6 in this memo:

Method	Value	RFCs
IEEE EUI-64	0x01	this memo
CGAs	0x02	this memo
Temporary	0x03	this memo
Stable, privacy	0x04	this memo

9. Acknowledgements

Valuable comments from Bernie Volz are appreciated.

10. References

10.1. Normative References

[dhcp_cga] Jiang, S., "Configuring Cryptographically Generated Addresses (CGA) using DHCPv6", 2009.

[dhcp_slaac_problem] Liu, B., "DHCPv6/SLAAC Interaction Problems on Address and DNS Configuration", 2016.

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

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <http://www.rfc-editor.org/info/rfc2464>.

[RFC3041] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, DOI 10.17487/RFC3041, January 2001, <http://www.rfc-editor.org/info/rfc3041>.

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

[RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <http://www.rfc-editor.org/info/rfc3971>.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, DOI 10.17487/RFC3972, March 2005, <http://www.rfc-editor.org/info/rfc3972>.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <http://www.rfc-editor.org/info/rfc4291>.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, DOI 10.17487/RFC4429, April 2006, <http://www.rfc-editor.org/info/rfc4429>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <http://www.rfc-editor.org/info/rfc4443>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

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

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

[RFC6957] Costa, F., Combes, J-M., Ed., Pougnard, X., and H. Li, "Duplicate Address Detection Proxy", RFC 6957, DOI 10.17487/RFC6957, June 2013, <http://www.rfc-editor.org/info/rfc6957>.

[RFC7037] Yeh, L. and M. Boucadair, "RADIUS Option for the DHCPv6 Relay Agent", RFC 7037, DOI 10.17487/RFC7037, October 2013, <http://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, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc7943>.

[secure_dhcpv6] Jiang, S., "Secure DHCPv6", 2016.

10.2. Informative References

[DOMINATION] Mad Dominators, Inc., "Ultimate Plan for Taking Over the World", 1984, <http://www.example.com/dominator.html>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <http://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7749] Reschke, J., "The "xml2rfc" Version 2 Vocabulary", RFC 7749, DOI 10.17487/RFC7749, February 2016, <http://www.rfc-editor.org/info/rfc7749>.

Appendix A. Additional Stuff

This becomes an Appendix.

Authors' Addresses

Gang Ren
Tsinghua University
Beijing
CN

Phone: +86-010 6260 3227
Email: rengang@cernet.edu.cn

Lin He
Tsinghua University
Beijing
CN

Email: he-l14@mails.tsinghua.edu.cn

Ying Liu
Tsinghua University
Beijing
CN

Email: liuying@cernet.edu.cn