Dynamic Host Configuration Working   D. Hankins
Group   Google
Internet-Draft   T. Mrugalski
Updates: 3315 (if approved)   M. Siodelski
Intended status: BCP   ISC
Expires: March 22, 2014   S. Jiang
                                   Huawei Technologies Co., Ltd
                                                     S. Krishnan
                                                        Ericsson
                                              September 18, 2013

Guidelines for Creating New DHCPv6 Options
draft-ietf-dhc-option-guidelines-14

Abstract

   This document provides guidance to prospective DHCPv6 Option
   developers to help them creating option formats that are easily
   adoptable by existing DHCPv6 software. This document updates
   RFC3315.

Status of this Memo

Copyright Notice

Table of Contents

1. 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 RFC 2119 [RFC2119].

2. Introduction

Most protocol developers ask themselves if a protocol will work, or
work efficiently. These are important questions, but another less
frequently considered question is whether the proposed protocol
presents itself needless barriers to adoption by deployed software.

DHCPv6 [RFC3315] software implementors are not merely faced with the
task of handling a given option's format on the wire. The option
must fit into every stage of the system's process, starting with the
user interface used to enter the configuration up to the machine
interfaces where configuration is ultimately consumed.

Another frequently overlooked aspect of rapid adoption is whether the
option requires operators to be intimately familiar with the option's
internal format in order to use it? Most DHCPv6 software provides a
facility for handling unknown options at the time of publication.
The handling of such options usually needs to be manually configured
by the operator. But if doing so requires extensive reading (more
than can be covered in a simple FAQ for example), it inhibits
adoption.

So although a given solution would work, and might even be space,
time, or aesthetically optimal, a given option is presented with a
series of ever-worsening challenges to be adopted:

o If it doesn't fit neatly into existing config files.

o If it requires source code changes to be adopted, and hence
   upgrades of deployed software.

o If it does not share its deployment fate in a general manner with
   other options, standing alone in requiring code changes or
   reworking configuration file syntaxes.

o If the option would work well in the particular deployment
   environment the proponents currently envision, but has equally
   valid uses in some other environment where the proposed option
   format would fail or would produce inconsistent results.

There are many things DHCPv6 option creators can do to avoid the

pitfalls in this list entirely, or failing that, to make software
implementors lives easier and improve its chances for widespread
adoption.

3. **When to Use DHCPv6**

Principally, DHCPv6 carries configuration parameters for its clients.
Any knob, dial, slider, or checkbox on the client system, such as "my
domain name servers", "my hostname", or even "my shutdown
temperature" are candidates for being configured by DHCPv6.

The presence of such a knob isn't enough, because DHCPv6 also
presents the extension of an administrative domain - the operator of
the network to which the client is currently attached. Someone runs
not only the local switching network infrastructure that the client
is directly (or wirelessly) attached to, but the various methods of
accessing the external Internet via local assist services that the
network must also provide (such as domain name servers, or routers).
This means that, even if a configuration parameter can potentially
delivered by DHCPv6, it is necessary to evaluate whether it is
reasonable for this parameter to be under the control of the
administrator of whatever network a client is attached to at any
given time.

Note that the client is not required to configure any of these values
received via DHCPv6 (e.g., due to having these values locally
configured by its own administrator). But it needs to be noted that
overriding DHCPv6-provided values may cause the client to be denied
certain services in the network to which it has attached. The
possibility of having higher level of control over client node
configuration is one of the reasons that DHCPv6 is preferred in
enterprise networks.

4. **General Principles**

The primary guiding principle to follow in order to enhance an
option's adoptability is reuse. The option should be created in such
a way that does not require any new or special case software to
support. If old software currently deployed and in the field can
adopt the option through supplied configuration facilities then it's
fairly certain that new software can easily formally adopt it.

There are at least two classes of DHCPv6 options: simple options
which are provided explicitly to carry data from one side of the
DHCPv6 exchange to the other (such as nameservers, domain names, or
time servers), and a protocol class of options which require special

processing on the part of the DHCPv6 software or are used during
special processing (such as the Fully Qualified Domain Name (FQDN)
option [RFC4704]), and so forth; these options carry data that is the
result of a routine in some DHCPv6 software.

The guidelines laid out here should be applied in a relaxed manner
for the protocol class of options. Wherever special case code is
already required to adopt the DHCPv6 option, it is substantially more
reasonable to format the option in a less generic fashion, if there
are measurable benefits to doing so.

5. Reusing Other Options Formats

The easiest approach to manufacturing trivially deployable DHCPv6
Options is to assemble the option out of whatever common fragments
fit - possibly allowing a group of data elements to repeat to fill
the remaining space (if present) and so provide multiple values.
Place all fixed size values at the start of the option, and any
variable/indeterminate sized value at the tail end of the option.

This means that implementations will likely be able to reuse code
paths designed to support the other options.

There is a tradeoff between the adoptability of previously defined
option formats, and the advantages that new or specialized formats
can provide. In general, it is usually preferable to reuse
previously used option formats.

However, it isn't very practical to consider the bulk of DHCPv6
options already allocated, and consider which of those solve a
similar problem. So, the following list of common option format data
elements is provided as a shorthand. Please note that it is not
complete in terms of exampling every option format ever devised.

If more complex options are needed, those basic formats mentioned
here may be considered as primitives (or 'fragment types') that can
be used to build more complex formats. It should be noted that it is
often easier to implement two options with trivial formats than one
option with more complex format. That is not unconditional
requirement though. In some cases splitting one complex option into
two or more simple options introduces inter-option dependencies that
should be avoided. In such a case, it is usually better to keep one
complex option.

5.1. **Option with IPv6 addresses**

   This option format is used to carry one or many IPv6 addresses. In
   some cases the number of allowed address is limited (e.g. to one):

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           option-code          |           option-len          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                          ipv6-address                         |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                          ipv6-address                         |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                              ...                              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
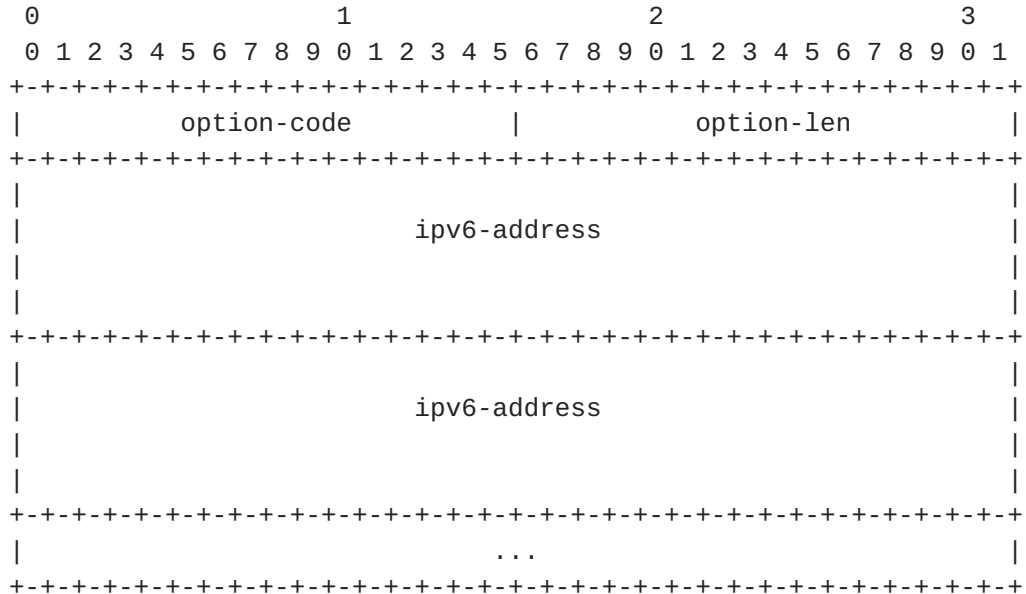
                  Figure 1: Option with IPv6 address

   Examples of use:

   o DHCPv6 server unicast address (a single address only) [RFC3315]

   o SIP Servers IPv6 Address List [RFC3319]

   o DNS Recursive Name Server [RFC3646]

   o NIS Servers [RFC3898]

   o SNTP Servers [RFC4075]

   o Broadcast and Multicast Service Controller IPv6 Address Option for
      DHCPv6 [RFC4280]

   o MIPv6 Home Agent Address [RFC6610] (a single address only)

   o NTP server [RFC5908] (a single address only)

   o NTP Multicast address [RFC5908] (a single address only)

5.2. Option with a single flag (boolean)

   Sometimes it is useful to convey a single flag that can either take
   on or off values. Instead of specifying an option with one bit of
   usable data and 7 bits of padding, it is better to define an option
   without any content. It is the presence or absence of the option
   that conveys the value. This approach has the additional benefit of
   absent option designating the default, i.e. administrator has to take
   explicit actions to deploy the opposite of the default value.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          option-code          |           option-len          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
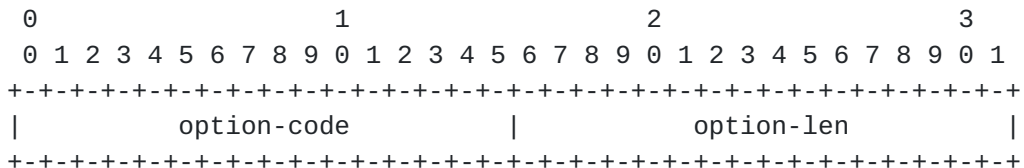
                   Figure 2: Option for conveying boolean

   Examples of use:

   o DHCPv6 rapid-commit [RFC3315]

5.3. Option with IPv6 prefix

   Sometimes there is a need to convey an IPv6 prefix. The information
   to be carried by such an option includes the 128-bit IPv6 prefix
   together with a length of this prefix taking values from 0 to 128.
   Using the simplest approach, the option could convey this data in two
   fixed length fields: one carrying prefix length, another carrying the
   prefix. However, in many cases /64 or shorter prefixes are used.
   This implies that the large part of the prefix data carried by the
   option would have its bits set to zero and would be unused. In order
   to avoid carrying unused data, it is recommended to store prefix in
   the variable length data field. The appropriate option format is
   defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |          option-length        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| prefix6-len  |              ipv6-prefix                        |
+-+-+-+-+-+-+-+-+-+              (variable length)               |
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
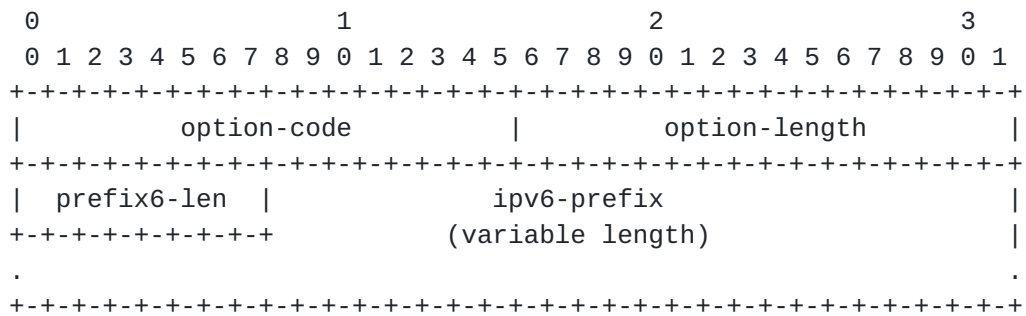
                    Figure 3: Option with IPv6 Prefix

   option-length is set to 1 + length of the IPv6 prefix.

   prefix6-len is one octet long and specifies the length in bits of the
   IPv6 prefix. Typically allowed values are 0 to 128.

   ipv6-prefix field is a variable length field that specifies the IPv6
   prefix. The length is (prefix6-len + 7) / 8. This field is padded
   with zero bits up to the nearest octet boundary when prefix6-len is
   not divisible by 8.

   Examples of use:

   o   Default Mapping Rule [I-D.ietf-softwire-map-dhcp]

   For example, the prefix 2001:db8::/60 would be encoded with an
   option-length of 9, prefix6-len would be set to 60, the ipv6-prefix
   would be 8 octets and would contain octets 20 01 0d b8 00 00 00 00.

   It should be noted that the IAPREFIX option defined by [RFC3633] uses
   a full length 16-octet prefix field. The concern about option length
   was not well understood at the time of its publication.

5.4. Option with 32-bit integer value

   This option format can be used to carry 32 bit-signed or unsigned
   integer value:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          32-bit-integer                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
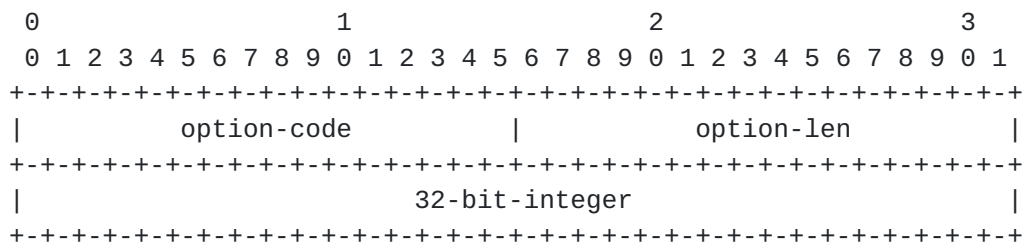
                 Figure 4: Option with 32-bit-integer value

   Examples of use:

   o Information Refresh Time [RFC4242]

5.5. Option with 16-bit integer value

   This option format can be used to carry 16-bit signed or unsigned
   integer values:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          option-code          |           option-len          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          16-bit-integer        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
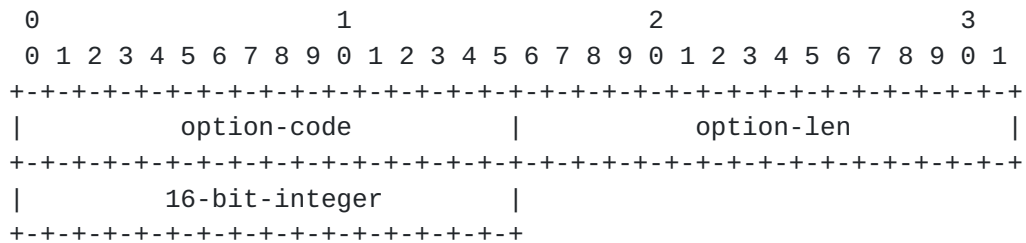
                 Figure 5: Option with 16-bit integer value

   Examples of use:

   o Elapsed Time [RFC3315]

5.6. Option with 8-bit integer value

   This option format can be used to carry 8-bit integer values:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          option-code          |           option-len          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | 8-bit-integer |
   +-+-+-+-+-+-+-+-+
```
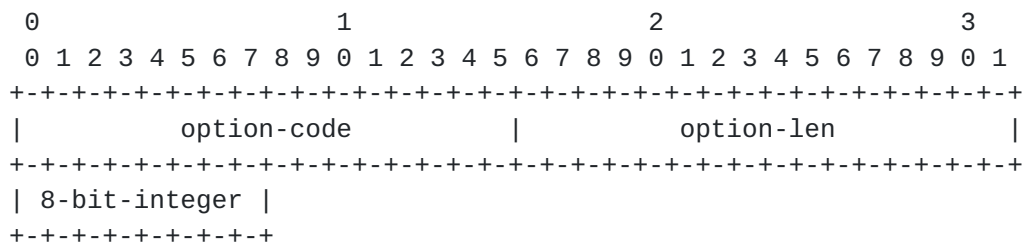
                  Figure 6: Option with 8-bit integer value

   Examples of use:

   o DHCPv6 Preference [RFC3315]

5.7. Option with URI

   A Uniform Resource Identifier (URI) [RFC3986] is a compact sequence
   of characters that identifies an abstract or physical resource. The
   term "Uniform Resource Locator" (URL) refers to the subset of URIs
   that, in addition to identifying a resource, provide a means of
   locating the resource by describing its primary access mechanism
   (e.g., its network "location"). This option format can be used to
   carry a single URI:

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           option-code         |           option-len          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 .                       URI (variable length)                   .
 |                           ...                                 |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
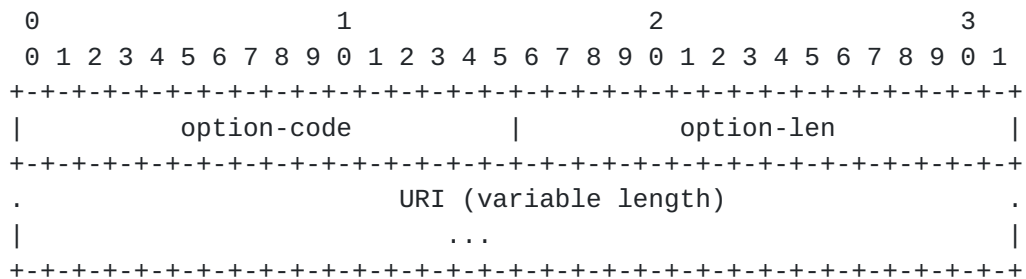
Figure 7: Option with URI

Examples of use:

o   Boot File URL [RFC5970]

An alternate encoding to support multiple URIs is available. An
option must be defined to use either the single URI format above or
the multiple URI format below depending on whether a single is always
sufficient or if multiple URIs are possible.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |           option-code         |           option-len          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 .                                                               .
 .                            uri-data                           .
 .                             . . .                             .
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: Option with multiple URIs

Each instance of the uri-data is formatted as follows:

```
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.,.-+-+-+-+-+-+
 |        uri-len                |            URI                |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-.,.-+-+-+-+-+-+
```

The uri-len is two octets long and specifies the length of the uri
data.

5.8. Option with Text String

A text string is a sequence of characters that have no semantics.
The encoding of the text string MUST be specified. Unless otherwise
specified, all text strings in newly defined options are expected to
be Unicode strings that are encoded using UTF-8 [RFC3629] in Net-

Unicode form [RFC5198]. Please note that all strings containing only
7 bit ASCII characters are also valid UTF-8 Net-Unicode strings.

If a data format has semantics other than just being text, it is not
a string. E.g., a FQDN is not a string, and a URI is also not a
string, because they have different semantics. A string must not
enclude any terminator (such as a null byte). This option format can
be used to carry a text string:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                            String                             .
|                             ...                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 9: Option with text string

Examples of use:

o Timezone Options for DHCPv6 [RFC4833]

An alternate encoding to support multiple text strings is available.
An option must be defined to use either the single text string format
above or the multiple text string format below depending on whether a
single is always sufficient or if multiple text strings are possible.
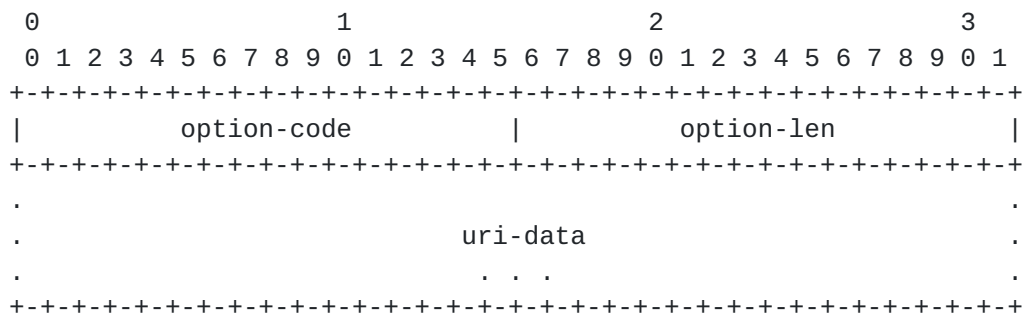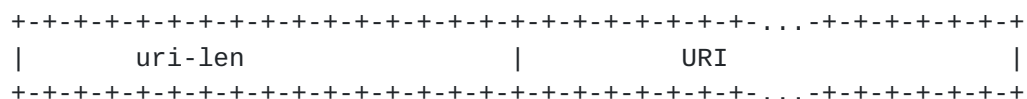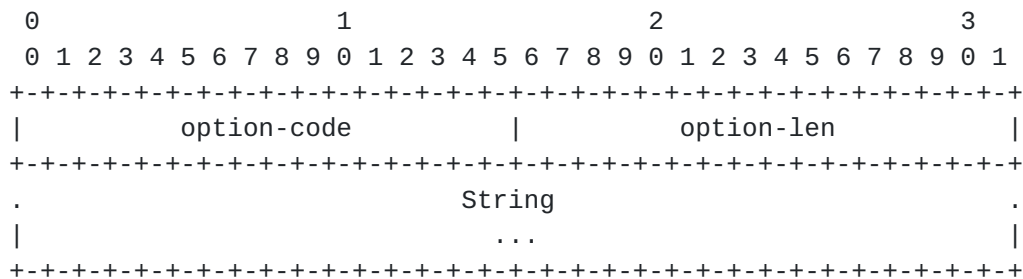
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          option-code          |          option-len           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
.                                                               .
.                           text-data                           .
.                            . . .                              .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
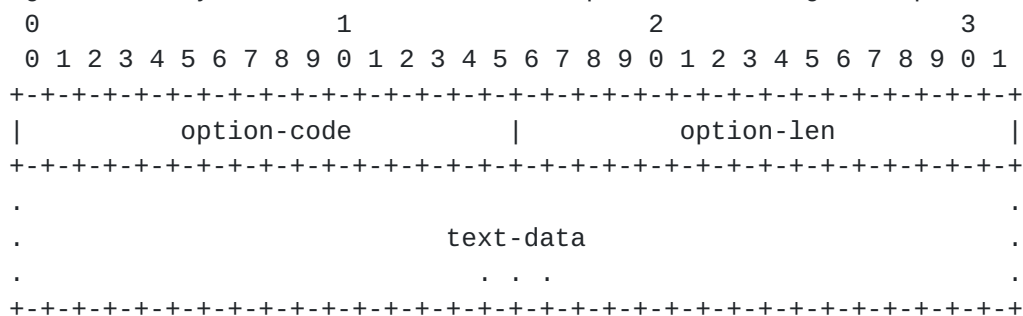
Figure 10: Option with multiple text strings

Each instance of the text-data is formatted as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+
|        text-len               |            String             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...-+-+-+-+-+-+
```
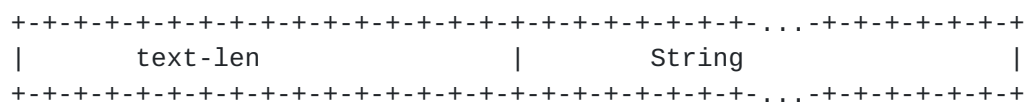
The text-len is two octets long and specifies the length of the
string.

5.9. **Option with variable length data**

   This option can be used to carry variable length data of any kind.
   Internal representation of carried data is option specific. Whenever
   this format is used by the new option being defined, the data
   encoding should be documented.

   This option format provides a lot of flexibility to pass data of
   almost any kind. Though, whenever possible it is highly recommended
   to use more specialized options, with field types better matching
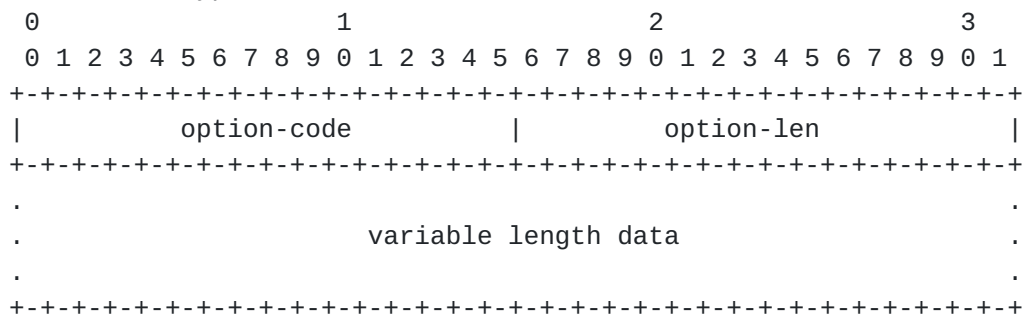   carried data types.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          option-code          |          option-len           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                     variable length data                     .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

                 Figure 11: Option with variable length data

   Examples of use:

   o Client Identifier [RFC3315]

   o Server Identifier [RFC3315]

5.10. **Option with DNS Wire Format Domain Name List**

   This option is used to carry 'domain search' lists or any host or
   domain name. It uses the same format as described in Section 5.9,
   but with the special data encoding, described in section 8 of
   [RFC3315]. This data encoding supports carrying multiple instances
   of hosts or domain names in a single option, by terminating each
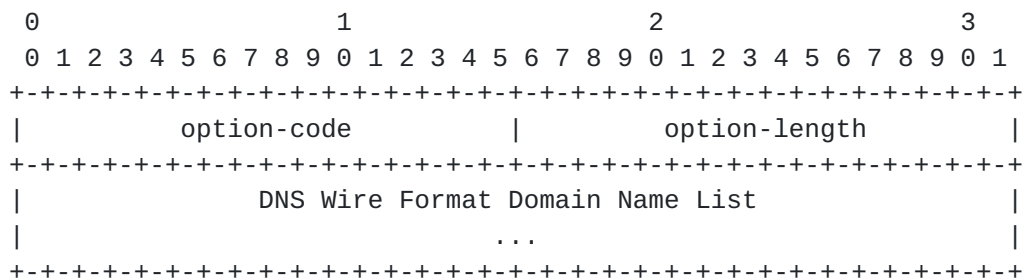   instance with the byte value of 0.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          option-code          |          option-length        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               DNS Wire Format Domain Name List                |
   |                             ...                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

              Figure 12: Option with DNS Wire Format Domain Name List

   Examples of use:

   o SIP Servers Domain Name List [RFC3319] (many domains)

   o NIS Domain Name (many domains) [RFC3898] (many domains)

   o DS-Lite AFTR location [RFC6334] (a single FQDN)

   o Home Network Identifier [RFC6610] (a single FQDN)

   o Home Agent FQDN [RFC6610] (a single FQDN)

6. Avoid Conditional Formatting

   Placing an octet at the start of the option which informs the
   software how to process the remaining octets of the option may appear
   simple to the casual observer. But the only conditional formatting
   methods that are in widespread use today are 'protocol' class
   options. Therefore conditional formatting requires new code to be
   written and complicates future interoperability should new
   conditional formats be added; and existing code has to ignore
   conditional format that it does not support.

7. Avoid Aliasing

   Options are said to be aliases of each other if they provide input to
   the same configuration parameter. A commonly proposed example is to
   configure the location of some new service ("my foo server") using a
   binary IP address, a domain name field, and an URL. This kind of
   aliasing is undesirable, and is not recommended.

   In this case, where three different formats are supposed, it more
   than triples the work of the software involved, requiring support for
   not merely one format, but support to produce and digest all three.
   Furthermore, code development and testing must cover all possible
   combinations of defined formats. Since clients cannot predict what
   values the server will provide, they must request all formats. So in
   the case where the server is configured with all formats, DHCPv6
   message bandwidth is wasted on option contents that are redundant.
   Also, the DHCPv6 option number space is wasted, as three new option
   codes are required, rather than one.

   It also becomes unclear which types of values are mandatory, and how
   configuring some of the options may influence the others. For
   example, if an operator configures the URL only, should the server
   synthesize a domain name and IP address?

A single configuration value on a host is probably presented to the operator (or other software on the machine) in a single field or channel. If that channel has a natural format, then any alternative formats merely make more work for intervening software in providing conversions.

So the best advice is to choose the one method that best fulfills the requirements, be that for simplicity (such as with an IP address and port pair), late binding (such as with DNS), or completeness (such as with a URL).

8. Choosing between FQDN and address

Some parameters may be specified as FQDN or an address. It is not allowed to define both option types at the same time (see section Section 7), so one of them must be chosen. This section is intended to help make an informed decision in that regard.

On the specific subject of desiring to configure a value using a FQDN instead of a binary IP address, note that most DHCPv6 server implementations will happily accept a Domain Name entered by the administrator, and use DNS resolution to render binary IP addresses in DHCPv6 replies to clients. Consequently, consider the extra packet overhead incurred on the client's end to perform DNS resolution itself. The client may be operating on a battery and packet transmission is a non-trivial use of power, and the extra RTT delays the client must endure before the service is configured are at least two factors to consider in making a decision on format.

Unless there are specific reasons to do otherwise, address should be used. It is simpler to use, its validation is trivial (length of 16 constitutes a valid option), is explicit and does not allow any ambiguity. It is faster (does not require extra resolution efforts), so it is more efficient, which can be especially important for energy restricted devices.

FQDN options are discouraged for options intended to configure hosts, because hosts may have multiple provisioning domains (see Section 12), and may get a different answer from the DNS depending on the provisioning domain. This is particularly a problem when the normal expected use of the option makes sense with private DNS zone(s), as might be the case with a corporate VPN.

FQDN does require a resolution into an actual address. This implies the question when the FQDN resolution should be taken. There are a couple of possible answers: a) by the server, when it is started, b) by the server, when it is about to send an option, c) by the client,

   immediately after receiving an option, d) by the client, when the
   content of the option is actually consumed. For a), b) and possibly
   c), the option should really convey an address, not FQDN. The only
   real incentive to use FQDN is case d). It is the only case that
   allows possible changes in the DNS to be picked up by clients.

   FQDN imposes a number of additional failure modes and issues that
   should be dealt with:

   1. The client must have a knowledge about available DNS servers.
       That typically means that option DNS_SERVERS is mandatory. This
       should be mentioned in the draft that defines new option. It is
       possible that the server will return FQDN option, but not the DNS
       Servers option. There should be a brief discussion about it;

   2. The DNS may not be reachable;

   3. DNS may be available, but may not have appropriate information
       (e.g. no AAAA records for specified FQDN);

   4. Address family must be specified (A, AAAA or any);

   5. What should the client do if there are multiple records available
       (use only the first one, use all, use one and switch to the
       second if the first fails for whatever reason, etc.);

   6. Multi-homed devices may be connected to different administrative
       domains with each domain providing different information in DNS
       (e.g. an enterprise network exposing private domains). Client
       may send DNS queries to a different DNS server;

   7. It should be mentioned if Internationalized Domain Names are
       allowed. If they are, what kind of DNS option encoding should be
       specified.

9. Encapsulated options in DHCPv6

   Most options are conveyed in a DHCPv6 message directly. Although
   there is no codified normative language for such options, they are
   often referred to as top-level options. Many options may include
   other options. Such inner options are often referred to as
   encapsulated or nested options. Those options are sometimes called
   sub-options, but this term actually means something else, and
   therefore should never be used to describe encapsulated options. It
   is recommended to use term "encapsulated" as this terminology is used
   in [RFC3315]. The difference between encapsulated and sub-options
   are that the former uses normal DHCPv6 option numbers, while the

latter uses option number space specific to a given parent option.
It should be noted that, contrary to DHCPv4, there is no shortage of
option numbers. Therefore almost all options share a common option
space. For example option type 1 meant different things in DHCPv4,
depending if it was located in top-level or inside of Relay Agent
Information option. There is no such ambiguity in DHCPv6 (with the
unfortunate exception of [RFC5908],which was published without
following the advice provided during the DHC working group review,
and contains many errors. [RFC5908] SHOULD NOT under any
circumstances be used as a template for future DHCP option
definitions.

From the implementation perspective, it is easier to implement
encapsulated options rather than sub-options, as the implementers do
not have to deal with separate option spaces and can use the same
buffer parser in several places throughout the code.

Such encapsulation is not limited to one level. There is at least
one defined option that is encapsulated twice: Identity Association
for Prefix Delegation (IA_PD, defined in [RFC3633], section 9)
conveys IA Prefix (IAPREFIX, defined in [RFC3633], section 10). Such
delegated prefix may contain an excluded prefix range that is
represented by PD_EXCLUDE option that is conveyed as encapsulated
inside IAPREFIX (PD_EXCLUDE, defined in [RFC6603]). It seems awkward
to refer to such options as sub-sub-option or doubly encapsulated
option, therefore "encapsulated option" term is typically used,
regardless of the nesting level.

When defining a DHCP-based configuration mechanism for a protocol
that requires something more complex than a single option, it may be
tempting to group configuration values using sub-options. That
should preferably be avoided, as it increases complexity of the
parser. It is much easier, faster and less error prone to parse a
large number of options on a single (top-level) scope, than parse
options on several scopes. The use of sub-options should be avoided
as much as possible, but it is better to use sub-options rather than
conditional formatting.

It should be noted that currently there is no clear way defined for
requesting sub-options. Most known implementations are simply using
top-level ORO for requesting both top-level options and encapsulated
options.

10. Additional States Considered Harmful

DHCP is a protocol designed for provisioning clients. Less
experienced protocol designers often assume that it is easy to define

an option that will convey a different parameter for each client in a
network. Such problems arose during designs of MAP
[I-D.ietf-softwire-map-dhcp] and 4rd [I-D.ietf-softwire-4rd]. While
it would be easier for provisioned clients to get ready to use per-
client option values, such requirement puts exceedingly large loads
on the server side. The new extensions may introduce new
implementation complexity and additional database state on the
server. Alternatives should be considered, if possible. As an
example, [I-D.ietf-softwire-map-dhcp] was designed in a way that all
clients are provisioned with the same set of MAP options and each
provisioned client uses its unique address and delegated prefix to
generate client-specific information. Such a solution does not
introduce any additional state for the server and therefore scales
better.

It also should be noted that contrary to DHCPv4, DHCPv6 keeps several
timers for renewals. Each IA_NA (addresses) and IA_PD (prefixes)
contains T1 and T2 timers that designate time after which client will
initiate renewal. Those timers apply only to its own IA containers.
Refreshing other parameters should be initiated after a time
specified in the Information Refresh Time Option (defined in
[RFC4242]), carried in the Reply message and returned in response to
Information-Request message. Introducing additional timers make
deployment unnecessarily complex and SHOULD be avoided.

11. Configuration changes occur at fixed times

In general, DHCPv6 clients only refresh configuration data from the
DHCP server when the T1 timer expires. Although there is a
RECONFIGURE mechanism that allows a DHCP server to request that
clients initiate reconfiguration, support for this mechanism is
optional and cannot be relied upon.

Even when DHCP clients refresh their configuration information, not
all consumers of DHCP-sourced configuration data notice these
changes. For instance, if a server is started using parameters
received in an early DHCP transaction, but does not check for updates
from DHCP, it may well continue to use the same parameter
indefinitely. There are a few operating systems that take care of
reconfiguring services when the client moves to a new network(e.g.
based on mechanisms like [RFC4436], [RFC4957] or [RFC6059]), but it's
worth bearing in mind that a renew may not always result in the
client taking up new configuration information that it receives.

In light of the above, when designing an option you should take into
consideration the fact that your option may hold stale data that will
only be updated at an arbitrary time in the future.

12. Multiple provisioning domains

In some cases there could be more than one DHCPv6 server on a link,
with each providing a different set of parameters. One notable
example of such a case is a home network with a connection to two
independent ISPs.

The DHCPv6 protocol specification does not provide clear advice on
how to handle multiple provisioning sources. Although [RFC3315]
states that a client that receives more than one ADVERTISE message,
may respond to one or more of them, such capability has not been
observed in existing implementations. Existing clients will pick one
server and will continue configuration process with that server,
ignoring all other servers.

In addition, a node that acts as a DHCPv6 client may be connected to
more than one physical network. In this case, it will in most cases
operate a separate DHCP client state machine on each interface,
acquiring different, possibly conflicting information through each.
This information will not be acquired in any synchronized way.

Existing nodes cannot be assumed to systematically segregate
configuration information on the basis of its source; as a result, it
is quite possible that a node may receive an FQDN on one network
interface, but do the DNS resolution on a different network
interface, using different DNS servers. As a consequence, DNS
resolution done by the DHCP server is more likely to behave
predictably than DNS resolution done on a multi-interface or multi-
homed client.

This is a generic DHCP protocol issue and should not be dealt within
each option separately. This issue is better dealt with using a
protocol-level solution and fixing this problem should not be
attempted on a per option basis. Work is ongoing in the IETF to
provide a systematic solution to this problem.

13. Chartering Requirements and Advice for Responsible ADs

Adding a simple DHCP option is straightforward, and generally
something that any working group can do, perhaps with some help from
designated DHCP experts. However, when new fragment types need to be
devised, this requires the attention of DHCP experts, and should not
be done in a working group that doesn't have a quorum of such
experts. This is true whether the new fragment type has the same
structure as an existing fragment type, but has different semantics.
It is equally true when the new format has a new structure.

   Responsible Area Directors for working groups that wish to add a work
   item to a working group charter to define a new DHCP option should
   get clarity from the working group as to whether the new option is a
   simple DHCP option with no new fragment type or new fragment
   semantics, or whether it in fact will require new fragment types. A
   working group charter item should explicitly state which of these two
   types is required; if it is not known at the time of chartering, the
   charter should state that the working group will study the question
   and recharter or seek help elsewhere if a new fragment type is to be
   defined.

   If a working group needs a new fragment type, it is preferable to
   seek out a working group whose members already have sufficient
   expertise to evaluate the new work and try to come up with a new
   format that generalizes well and can be reused, rather than a single-
   use fragment type. If such a working group is available, the work
   should be chartered in that working group as a separate draft that
   documents the new fragment type. The working group that needs the
   new fragment type can then define their new option referencing the
   new fragment type document. This work can generally be done in
   parallel so as not to delay the process significantly.

   In the event that there is no working group with DHCP expertise that
   can define the new fragment type, the responsible AD should seek out
   help from known DHCP experts within the IETF to provide advice and
   frequent early review as the working group defines the new fragment
   type. The new fragment type should still be done in a separate
   document, even if it's done in the same working group, so as to
   foster reuse of the new fragment type. The responsible AD should
   work with the working group chairs and designated DHCP experts to
   ensure that new fragment type document has in fact been carefully
   reviewed by the experts and appears satisfactory.

   Responsible area directors for working groups that are considering
   defining options that actually update the DHCP protocol, as opposed
   to simple options, should go through a process similar to that
   described above when trying to determine where to do the work. Under
   no circumstances should a working group be given a charter
   deliverable to define a new DHCP option, and then on the basis of
   that charter item actually make updates to the DHCP protocol.

14. Considerations for Creating New Formats

   When defining new options, one specific consideration to evaluate is
   whether or not options of a similar format would need to have
   multiple or single values encoded (whatever differs from the current
   option), and how that might be accomplished in a similar format.

When defining a new option, it is best to synthesize the option
format using fragment types already in use. However, in some cases
there may be no fragment type that accomplishes the intended purpose.

The matter of size considerations and option order are further
discussed in Section 15 and Section 17.

15. Option Size

DHCPv6 [RFC3315] allows for packet sizes up to 64KB. First, through
its use of link-local addresses, it avoids many of the deployment
problems that plague DHCPv4, and is actually an UDP over IPv6 based
protocol (compared to DHCPv4, which is mostly UDP over IPv4 protocol,
but with layer 2 hacks). Second, RFC 3315 explicitly refers readers
to RFC 2460 Section 5, which describes an MTU of 1280 octets and a
minimum fragment reassembly of 1500 octets. It's feasible to suggest
that DHCPv6 is capable of having larger options deployed over it, and
at least no common upper limit is yet known to have been encoded by
its implementors. It is not really possible to describe a fixed
limit that cleanly divides workable option sizes from those that are
too big.

It is advantageous to prefer option formats which contain the desired
information in the smallest form factor that satisfies the
requirements. Common sense still applies here. It is better to
split distinct values into separate octets rather than propose overly
complex bit shifting operations to save several bits (or even an
octet or two) that would be padded to the next octet boundary anyway.

DHCPv6 does allow for multiple instances of a given option, and they
are treated as distinct values following the defined format, however
this feature is generally preferred to be restricted to protocol
class features (such as the IA_* series of options). In such cases,
it is better to define an option as an array if it is possible. It
is recommended to clarify (with normative language) whether a given
DHCPv6 option may appear once or multiple times. The default
assumption is only once.

In general, if a lot of data needs to be configured (i.e. large
option lengths), DHCPv6 may not be the best choice to deliver such
configuration information and SHOULD simply be used to deliver an URI
that specifies how to obtain the actual configuration information.

16. Singleton options

Although [RFC3315] states that each option type MAY appear more than

once, the original idea was that multiple instances are reserved for
stateful options, like IA_NA or IA_PD. For most other options it is
usually expected that they will appear at most once. Such options
are called singleton options. Sadly, RFCs have often failed to
clearly specify whether a given option can appear more than once or
not. Documents that define new options SHOULD state whether these
options are singletons or not. Unless otherwise specified, newly
defined options are considered to be singletons.

When deciding whether a single or multiple option instances are
allowed in a message, take into consideration how the content of the
option will be used. Depending on the service being configured it
may or may not make sense to have multiple values configured. If
multiple values make sense, it is better to explicitly allow that by
using option format that allows multiple values within one option
instance.

Allowing multiple option instances often leads to confusion.
Consider the following example. Basic DS-Lite architecture assumes
that the B4 element (DHCPv6 client) will receive AFTR option and
establish a single tunnel to configured tunnel termination point
(AFTR). During standardization process of [RFC6334] there was a
discussion whether multiple instances of DS-Lite tunnel option should
be allowed. This created an unfounded expectation that the clients
receiving multiple instances of the option will somehow know when one
tunnel endpoint goes off-line and do some sort of failover between
other values provided in other instances of the AFTR option. Others
assumed that if there are multiple options, the client will somehow
do a load balancing between provided tunnel endpoints. Neither
failover nor load balancing was defined for DS-Lite architecture, so
it caused confusion. It was eventually decided to allow only one
instance of the AFTR option.

17. Option Order

Option order, either the order among many DHCPv6 options or the order
of multiple instances of the same option, SHOULD NOT be significant
and MUST NOT be assumed.

As there is no explicit order for multiple instance of the same
option, an option definition SHOULD instead restrict ordering by
using a single option that contains ordered fields.

18. Relay Options

In DHCPv4, all relay options are organized as sub-options within DHCP

Relay Agent Information Option[RFC3046]. And an independent number
space called "DHCP Relay Agent Sub-options" is maintained by IANA.
Different from DHCPv4, in DHCPv6, Relay options are defined in the
same way as client/server options, and they too use the same number
space as client/server options. Future DHCPv6 Relay options MUST be
allocated from this single DHCPv6 Option number space.

E.g. the Relay-Supplied Options Option [RFC6422] may also contain
some DHCPv6 options as permitted, such as the EAP Re-authentication
Protocol (ERP) Local Domain Name DHCPv6 Option [RFC6440].

19. Clients Request their Options

The DHCPv6 Option Request Option (OPTION_ORO) [RFC3315], is an option
that serves two purposes - to inform the server what options the
client supports and to inform what options the client is willing to
consume.

For some options, such as the options required for the functioning of
the DHCPv6 protocol itself, it doesn't make sense to require that
they be explicitly requested using the Option Request Option. In all
other cases, it is prudent to assume that any new option must be
present on the relevant option request list if the client desires to
receive it.

It is tempting to add text that requires the client to include a new
option in Option Request Option list, similar to this text: "Clients
MUST place the foo option code on the Option Request Option list,
clients MAY include option foo in their packets as hints for the
server as values the desire, and servers MUST include option foo when
the client requested it (and the server has been so configured)".
Such text is discouraged as there are several issues with it. First,
it assumes that client implementation that supports a given option
will always want to use it. This is not true. The second and more
important reason is that such text essentially duplicates mechanism
already defined in [RFC3315]. It is better to simply refer to the
existing mechanism rather than define it again. See Section 21 for
proposed examples on how to do that.

Creators of DHCPv6 options cannot not assume special ordering of
options either as they appear in the option request option, or as
they appear within the packet. Although it is reasonable to expect
that options will be processed in the order they appear in ORO,
server software is not required to sort DHCPv6 options into the same
order in reply messages.

It should also be noted that options values are never aligned within

the DHCP packet, even the option code and option length may appear on
odd byte boundaries.

20. Transition Technologies

Transition from IPv4 to IPv6 is progressing. Many transition
technologies are proposed to speed it up. As a natural consequence
there are also DHCP options proposed to provision those proposals.
The inevitable question is whether the required parameters should be
delivered over DHCPv4 or DHCPv6. Authors often don't give much
thought about it and simply pick DHCPv6 without realizing the
consequences. IPv6 is expected to stay with us for many decades, and
so is DHCPv6. There is no mechanism available to deprecate an option
in DHCPv6, so any options defined will stay with us as long as DHCPv6
protocol itself. It seems likely that such options defined to
transition from IPv4 will outlive IPv4 by many decades. From that
perspective it is better to implement provisioning of the transition
technologies in DHCPv4, which will be obsoleted together with IPv4.

When the network infrastructure becomes IPv6-only, the support for
IPv4-only nodes may still be needed. In such a scenario, a mechanism
for providing IPv4 configuration information over IPv6-only networks
such as [I-D.ietf-dhc-v4configuration] may be needed.

21. Recommended sections in the new document

There are three major entities in DHCPv6 protocol: server, relay
agent, and client. It is very helpful for implementers to include
separate sections that describe operation for those three major
entities. Even when a given entity does not participate, it is
useful to have a very short section stating that it must not send a
given option and must ignore it when received.

There is also a separate entity called requestor, which is a special
client-like type that participates in leasequery protocol [RFC5007]
and [RFC5460]. A similar section for the requestor is not required,
unless the new option has anything to do with requestor (or it is
likely that the reader may think that is has). It should be noted
that while in the majority of deployments, requestor is co-located
with relay agent, those are two separate entities from the protocol
perspective and they may be used separately. There are stand-alone
requestor implementations available.

The following sections include proposed text for such sections. That
text is not required to appear, but it is appropriate in most cases.
Additional or modified text specific to a given option is often

   required.

   Although requestor is somewhat uncommon functionality, its existence
   should be noted, especially when allowing or disallowing options to
   appear in certain message or being sent by certain entities.
   Additional message types may appear in the future, besides types
   defined in [RFC3315]. Therefore authors are encouraged to
   familiarize themselves with a list of currently defined DHCPv6
   messages available on IANA website [iana].

   Typically new options are requested by clients and assigned by the
   server, so there is no specific relay behavior. Nevertheless it is
   good to include a section for relay agent behavior and simply state
   that there are no additional requirements for relays. The same
   applies for client behavior if the options are to be exchanged
   between relay and server.

   Sections that contain option definitions MUST include formal
   verification procedure. Often it is very simple, e.g. option that
   conveys IPv6 address must be exactly 16 bytes long, but sometimes the
   rules are more complex. It is recommeded to refer to existing
   documents (e.g. section 8 of RFC3315 for domain name encoding) rather
   than trying to repeat such rules.

21.1. DHCPv6 Client Behavior Text

   Clients MAY request option foo, as defined in [RFC3315], sections
   17.1.1, 18.1.1, 18.1.3, 18.1.4, 18.1.5 and 22.7. As a convenience to
   the reader, we mention here that the client includes requested option
   codes in Option Request Option.

   Optional text (if client's hints make sense): Client also MAY include
   option foo in its SOLICIT, REQUEST, RENEW, REBIND and INFORMATION-
   REQUEST messages as a hint for the server regarding preferred option
   values.

   Optional text (if the option contains FQDN): If the client requests
   an option that conveys an FQDN, it is expected that the contents of
   that option will be resolved using DNS. Hence the following text may
   be useful: Clients that request option foo SHOULD also request option
   OPTION_DNS_SERVERS specified in [RFC3646].

   Clients MUST discard option foo if it is invalid (i.e. did not pass
   validation steps defined in Section X.Y).

   Optional text (if option foo in expected to be exchanged between
   relays and servers): Option foo is exchanged between relays and
   servers only. Clients are not aware of the usage of option foo.

   Clients MUST ignore received option foo.

21.2. DHCPv6 Server Behavior Text

   Sections 17.2.2 and 18.2 of [RFC3315] govern server operation in
   regards to option assignment. As a convenience to the reader, we
   mention here that the server will send option foo only if configured
   with specific values for foo and the client requested it.

   Optional text: Option foo is a singleton. Servers MUST NOT send more
   than one instance of foo option.

   Optional text (if server is never supposed to receive option foo):
   Servers MUST ignore incoming foo option.

21.3. DHCPv6 Relay Agent Behavior Text

   It's never appropriate for a relay agent to add options to a message
   heading toward the client, and relay agents don't actually construct
   Relay-Reply messages anyway.

   Optional text (if foo option is exchanged between clients and server
   or between requestors and servers): There are no additional
   requirements for relays.

   Optional text (if relays are expected to insert or consume option
   foo): Relay agents MAY include option foo in a Relay-Forw when
   forwarding packets from clients to the servers.

22. Should the new document update existing RFCs?

   Authors often ask themselves a question whether their proposal
   updates exist RFCs, especially 3315. In April 2013 there were about
   80 options defined. Had all documents that defined them also updated
   RFC3315, comprehension of such a document set would be extremely
   difficult. It should be noted that "extends" and "updates" are two
   very different verbs. If a new draft defines a new option that
   clients request and servers provide, it merely extends current
   standards, so "updates 3315" is not required in the new document
   header. On the other hand, if a new document replaces or modifies
   existing behavior, it should be noted that it updates the other
   document. For example, [RFC6644] clearly updates [RFC3315] as it
   replaces existing with new text.

23. Security Considerations

   DHCPv6 does have an Authentication mechanism ([RFC3315]) that makes
   it possible for DHCPv6 software to discriminate between authentic
   endpoints and man-in-the-middle. Other authentication mechanisms may
   optionally be deployed.

   So, while creating a new option, it is prudent to assume that the
   DHCPv6 packet contents are always transmitted in the clear, and
   actual production use of the software will probably be vulnerable at
   least to man-in-the-middle attacks from within the network, even
   where the network itself is protected from external attacks by
   firewalls. In particular, some DHCPv6 message exchanges are
   transmitted to multicast addresses that are likely broadcast anyway.

   If an option is of a specific fixed length, it is useful to remind
   the implementer of the option data's full length. This is easily
   done by declaring the specific value of the 'length' tag of the
   option. This helps to gently remind implementers to validate option
   length before digesting them into likewise fixed length regions of
   memory or stack.

   If an option may be of variable size (such as having indeterminate
   length fields, such as domain names or text strings), it is advisable
   to explicitly remind the implementor to be aware of the potential for
   long options. Either define a reasonable upper limit (and suggest
   validating it), or explicitly remind the implementor that an option
   may be exceptionally long (to be prepared to handle errors rather
   than truncate values).

   For some option contents, out of bound values may be used to breach
   security. An IP address field might be made to carry a loopback
   address, or local multicast address, and depending on the protocol
   this may lead to undesirable results. A domain name field may be
   filled with contrived contents that exceed the limitations placed
   upon domain name formatting - as this value is possibly delivered to
   "internal configuration" records of the system, it may be implicitly
   trusted without being validated.

   Authors of drafts defining new DHCP options are therefore strongly
   advised to explicitly define validation measures that recipients of
   such options are required to do before processing such options.
   However, validation measures already defined by RFC3315 or other
   specifications referenced by the new option document are redundant,
   and can introduce errors, so authors are equally strongly advised to
   refer to the base specification for any such validation language
   rather than copying it into the new specification.

24. IANA Considerations

   This document has no actions for IANA.

25. Acknowledgements

   Authors would like to thank Simon Perreault, Bernie Volz, Ted Lemon,
   Bud Millwood and Ralph Droms for their comments.

26. References

26.1. Normative References

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

   [RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C.,
              and M. Carney, "Dynamic Host Configuration Protocol for
              IPv6 (DHCPv6)", RFC 3315, July 2003.

26.2. Informative References

   [I-D.ietf-dhc-v4configuration]
              Rajtar, B. and I. Farrer, "Provisioning IPv4 Configuration
              Over IPv6 Only Networks",
              draft-ietf-dhc-v4configuration-01 (work in progress),
              May 2013.

   [I-D.ietf-softwire-4rd]
              Despres, R., Jiang, S., Penno, R., Lee, Y., Chen, G., and
              M. Chen, "IPv4 Residual Deployment via IPv6 - a Stateless
              Solution (4rd)", draft-ietf-softwire-4rd-06 (work in
              progress), July 2013.

   [I-D.ietf-softwire-map-dhcp]
              Mrugalski, T., Deng, X., Troan, O., Bao, C., Dec, W., and
              l. leaf.yeh.sdo@gmail.com, "DHCPv6 Options for
              configuration of Softwire Address and Port Mapped
              Clients", draft-ietf-softwire-map-dhcp-04 (work in
              progress), July 2013.

   [RFC3046] Patrick, M., "DHCP Relay Agent Information Option",
              RFC 3046, January 2001.

   [RFC3319] Schulzrinne, H. and B. Volz, "Dynamic Host Configuration
              Protocol (DHCPv6) Options for Session Initiation Protocol

                (SIP) Servers", RFC 3319, July 2003.

   [RFC3629]   Yergeau, F., "UTF-8, a transformation format of ISO
                10646", STD 63, RFC 3629, November 2003.

   [RFC3633]   Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic
                Host Configuration Protocol (DHCP) version 6", RFC 3633,
                December 2003.

   [RFC3646]   Droms, R., "DNS Configuration options for Dynamic Host
                Configuration Protocol for IPv6 (DHCPv6)", RFC 3646,
                December 2003.

   [RFC3898]   Kalusivalingam, V., "Network Information Service (NIS)
                Configuration Options for Dynamic Host Configuration
                Protocol for IPv6 (DHCPv6)", RFC 3898, October 2004.

   [RFC3986]   Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
                Resource Identifier (URI): Generic Syntax", STD 66,
                RFC 3986, January 2005.

   [RFC4075]   Kalusivalingam, V., "Simple Network Time Protocol (SNTP)
                Configuration Option for DHCPv6", RFC 4075, May 2005.

   [RFC4242]   Venaas, S., Chown, T., and B. Volz, "Information Refresh
                Time Option for Dynamic Host Configuration Protocol for
                IPv6 (DHCPv6)", RFC 4242, November 2005.

   [RFC4280]   Chowdhury, K., Yegani, P., and L. Madour, "Dynamic Host
                Configuration Protocol (DHCP) Options for Broadcast and
                Multicast Control Servers", RFC 4280, November 2005.

   [RFC4436]   Aboba, B., Carlson, J., and S. Cheshire, "Detecting
                Network Attachment in IPv4 (DNAv4)", RFC 4436, March 2006.

   [RFC4704]   Volz, B., "The Dynamic Host Configuration Protocol for
                IPv6 (DHCPv6) Client Fully Qualified Domain Name (FQDN)
                Option", RFC 4704, October 2006.

   [RFC4833]   Lear, E. and P. Eggert, "Timezone Options for DHCP",
                RFC 4833, April 2007.

   [RFC4957]   Krishnan, S., Montavont, N., Njedjou, E., Veerepalli, S.,
                and A. Yegin, "Link-Layer Event Notifications for
                Detecting Network Attachments", RFC 4957, August 2007.

   [RFC5007]   Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng,
                "DHCPv6 Leasequery", RFC 5007, September 2007.

   [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network
              Interchange", RFC 5198, March 2008.

   [RFC5460] Stapp, M., "DHCPv6 Bulk Leasequery", RFC 5460,
              February 2009.

   [RFC5908] Gayraud, R. and B. Lourdelet, "Network Time Protocol (NTP)
              Server Option for DHCPv6", RFC 5908, June 2010.

   [RFC5970] Huth, T., Freimann, J., Zimmer, V., and D. Thaler, "DHCPv6
              Options for Network Boot", RFC 5970, September 2010.

   [RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for
              Detecting Network Attachment in IPv6", RFC 6059,
              November 2010.

   [RFC6334] Hankins, D. and T. Mrugalski, "Dynamic Host Configuration
              Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite",
              RFC 6334, August 2011.

   [RFC6422] Lemon, T. and Q. Wu, "Relay-Supplied DHCP Options",
              RFC 6422, December 2011.

   [RFC6440] Zorn, G., Wu, Q., and Y. Wang, "The EAP Re-authentication
              Protocol (ERP) Local Domain Name DHCPv6 Option", RFC 6440,
              December 2011.

   [RFC6603] Korhonen, J., Savolainen, T., Krishnan, S., and O. Troan,
              "Prefix Exclude Option for DHCPv6-based Prefix
              Delegation", RFC 6603, May 2012.

   [RFC6610] Jang, H., Yegin, A., Chowdhury, K., Choi, J., and T.
              Lemon, "DHCP Options for Home Information Discovery in
              Mobile IPv6 (MIPv6)", RFC 6610, May 2012.

   [RFC6644] Evans, D., Droms, R., and S. Jiang, "Rebind Capability in
              DHCPv6 Reconfigure Messages", RFC 6644, July 2012.

   [iana]   IANA, "DHCPv6 parameters (IANA webpage)", November 2003,
              <http://www.iana.org/assignments/dhcpv6-parameters/>.

Authors' Addresses

   David W. Hankins
   Google, Inc.
   1600 Amphitheatre Parkway
   Mountain View, CA   94043
   USA

   Email: dhankins@google.com

   Tomek Mrugalski
   Internet Systems Consortium, Inc.
   950 Charter Street
   Redwood City, CA   94063
   USA

   Phone: +1 650 423 1345
   Email: tomasz.mrugalski@gmail.com

   Marcin Siodelski
   950 Charter Street
   Redwood City, CA   94063
   USA

   Phone: +1 650 423 1431
   Email: msiodelski@gmail.com

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

   Email: jiangsheng@huawei.com

   Suresh Krishnan
   Ericsson
   8400 Blvd Decarie
   Town of Mount Royal, Quebec
   Canada

   Email: suresh.krishnan@ericsson.com