Guidelines for Creating New DHCP Options
draft-ietf-dhc-option-guidelines-00

Status of this Memo

Copyright Notice

Abstract

This document seeks to provide guidance to prospective DHCP Option
authors, to help them in producing option formats that are easily
adoptable.

Table of Contents

[1]. **Introduction**

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

   DHCPv4 [1] and DHCPv6 [2] software implementors are not merely faced
   with the task of a given option's format on the wire. The option
   must 'fit' into every stage of the system's process, which includes
   user interface considerations. As an aide to understanding the
   potential implementation challenges of any new DHCP Option, one
   implementation's approach to tackling DHCP Option formats
   (Appendix A) has been included in an Appendix.

   Another, and more frequently overlooked, aspect of rapid adoption is
   wether or not the option would require operators to be intimately
   familiar with the option's internal format in order to make use of
   it. Most DHCP software provides a facility for "unknown options" at
   the time of publication to be configured by hand by an 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 going to have a
   rough time being adopted by deployed software if it requires code
   changes. A rougher time still, if it does not share its deployment
   fate in a general manner with other options of pressing need.

   There are many things DHCP option authors can do to form DHCP Options
   to make software implementors lives easier, and improve the chances
   that the Option is formally adopted in deployed software after it has
   been assigned an Option Code.

[2]. **When to Use DHCP**

   Principally, DHCP 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 DHCP.

   The presence of such a knob isn't enough, however, because
   secondarily, DHCP also presents the extension of an administrative
   domain - that of the systems 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 network must also
provide (such as domain name servers, or routers). This means taht
in addition to the existence of a configuration parameter, one must
also ask themselves if it is reasoanble for this parameter to be set
by the DHCP server operator.

Bear in mind that the client still reserves the right to over-ride or
ignore values received via DHCP (eg due to having a manually
configured value by its operator), and that at least one main use
case for DHCP is the corporate enterprise - so even if the local Net
Cafe is not a suitable source of this configuration, it is likely
that the client will at some point return to a network whose operator
is also the system's rightful master.

3. General Principles

The primary principle to follow in order to enhance an option's
adoptability is certainly simplification. But more specifically, to
create the option in such a way that it should 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
conveniences then it's fairly well assured that new software can
easily formally adopt it.

There are at least two classes of DHCP options. A bulk class of
options which are provided explicitly to carry data from one side of
the DHCP 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 DHCP software or are used during
special processing (such as the FQDN options ([3], [4]), DHCPv4
message type option [5], link selection options ([6], [7]), and so
forth).

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

4. Reusing Other Options

In DHCPv4, there are now nearly one hundred and thirty options, at
least as IETF standards, which might be used as an example. There is
also one handy document [5] containing many option definitions.

Although some may not like the way an old option that solves a

similar problem was approached, and it may waste space or processing
time or have ugly characteristics, it can usually be said that
duplicating that which has already been adopted has the greatest
chance of being adopted quickly and easily.

So it is preferrable to consider the bulk of DHCP options already
allocated, and consider which of those solve a similar problem. It
may even be that an option that solves the problem already exists.

But as there are so many options to choose from, this isn't entirely
practical. So, the following list of common option formats is
provided as a shorthand. Please note that it is not complete in
terms of exampling every option format ever devised...it is only a
list of option format fragments which are used in two or more
options.

                    Common Option Fragments

+---------------+-------+---+
| Fragment | Size | Types of Uses |
+---------------+-------+---+
ipv4-address	4	Default gateway, requested address,
		subnet mask [5], addresses of servers
		([5], [8], [9], [10], [11], [12]), as a
		component in a list of routes [13].
ipv6-address	16	DHCPv6 server unicast address [2],
		addresses of servers ([14], [15], [16],
		[17], [18]).
32-bit	4	Signed or unsigned varieties. Deprecated
integer		[19] use for timezone time offset [5].
		Other uses for host configuration values
		such as path mtu aging timeouts, arp
		cache timeouts, tcp keepalive intervals
		[5]. Also used by the DHCPv4 protocol
		for relative times, and times since
		epoch.
16-bit	2	Client configuration parameters, such as
integer		MTU, maximum datagram reassembly limits,
		the DHCPv4 maximum message size [5], or
		the elapsed time option [2] in DHCPv6.
8-bit integer	1	Used for host configuration parameters,
		such as the default IP TTL, default TCP
		TTL, NetBIOS node type [5]. Also used
		for protocol features, such as the DHCPv4
		Option Overload (as flags), DHCP Message
		Type (as an enumeration) or DHCPv6
		Preference [2].

NVT-Ascii Text	unlim	This is the kitchen sink of common fragments. Common uses are for filenames (such as TFTP paths), host or domain names (but this should be discouraged), or protocol features such as textual messages such as verbose error indicators. Since the size of this format cannot be determined (it is not NULL terminated), it consumes any remaining space in the option.
DNS Wire Format Domain Name List [20]	unlim	Presently used for 'domain search' lists in both DHCPv4 [21] and DHCPv6 [15], but also used in DHCPv6 for any host or domain name. A field formatted this way may have a determinate length if the number of root labels is limited, but use of this format as being a determinate length should be discouraged in DHCPv4, less so in DHCPv6.
'suboption' encapsulation	unlim	The Relay Agent Information Option [22], vendor options [5], Vendor Identified Vendor SubOptions ([23], [2]). Commonly used for situations where the full format cannot be known initially, such as where there seems to be some room for later protocol work to expand the amount of information carried, or where the full extent of data carried is defined in a private specification (such as with vendor options). Encapsulations do not use 'PAD' and 'END' options in DHCPv4, and there are no such options in DHCPv6, so this format also is of indeterminate length.

Table 1

One approach to manufacturing simple DHCP Options is to assemble the option out of whatever common fragments fit - possibly allowing one or more fragments 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 values at the tail end of the option. If there are more than one variable/ indeterminate length values, consider the use of multiple options or suboptions.

This estimates that implementations will be able to reuse code paths

   designed to support the other options.

5. **Conditional Formatting is Hard**

   Placing a byte at the start of the option which informs the software
   how to process the remaining bytes of the option may appear simple to
   the casual observer. But there are no such conditional formatting
   methods in existence today, so it must therefore require new, special
   case code, be written for this purpose.

   Wherever possible, used fixed length buffers to carry the information
   desired. Obviously, this becomes less possible as the fixed length
   buffer approaches large sizes, at least in DHCPv4, where DHCP packet
   space is limited.

6. **Aliasing**

   Providing multiple different formats of the same configuration
   information, such as an IP address, name, or URL, which are all
   intended to provide the same location information, is undesirable.

   In this case, where three different formats are supposed, it triples
   the work of the software involved, requiring support for not merely
   one format, but support to produce and digest all three. 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, DHCP option space is wasted on option contents that
   are redundant.

   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 parameter should have only one option to
   configure it. 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), late binding (such as with DNS), or completeness
   (such as with a URL).

7. **New Formats**

   If the Option simply will not fit into any existing work, the last
   recourse is to contrive a new format to fit.

When doing so, it is not enough to gauge wether or not the option
format will work in the context of the option presently being
considered. It is equally important to consider if the new format
might reasonably have any other uses, and if so, to create the option
with the foreknowledge that it may later become a common fragment.

One specific consideration to evaluate, is wether 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.

8. Option Size

DHCPv4 [1] options payload space is limited, as there are a number of
unaddressed deployment problems with DHCPv4 packet sizes. The end
result is that you should build your option to the assumption that
the packet will be no larger than 576 bytes. This means that the
options payload space will be 312 bytes, which you will have to share
with other options. This space can be extended by making use of
Option Overloading [5], which allows the use of the BOOTP FILE and
SNAME header fields for carrying DHCPv4 options (adding 192 bytes),
but these header fields will not be available for overloading if they
have been configured to carry a value.

DHCPv6 [2] carries a much more relaxed restriction, as it appears
ready and able to accept packet sizes up to 64KB, putting options
payload space at very nearly the same number (there are very few, and
small, header fields). But it is still undesirable to produce
fragments, and it's still very possible that the client's MTU is not
very large (or that client software is not prepared to retain a 64KB
buffer). So it is still best advice to design options to a ~500 byte
payload limitation.

This is easily accomplished by preferring option formats which
contain the desired information in the smallest form factor, in the
absence of other motivations. One example is to use a 4-octet IPv4
address rather than a fully qualified domain name. There may be
motivations to use the fully qualified domain name anyway, such as
externally supplied load balancers, or other protocol features.

When it is not possible to compress the configuration contents either
because of the number of optional parameters that must be identified,
or because it is expected that very large configurations are valid,
it may be preferrable to use a second stage configuration. Some
examples of this are to provide TFTP server and pathnames, or a URL,
which the client will load and process externally to the DHCP
protocol.

In the case where a DHCPv4 option may, or will, exceed 255 bytes in
length (and thus exceed the 'length' field's ability to contain it),
a DHCPv4 option will simply be fragmented into multiple options
within the packet. DHCP software processing these fragments will
concatenate them, in the order they appear as defined by RFC2131 [1],
prior to evaluating their contents. This is an important distinction
that is sometimes overlooked by authors - these multiple options do
not represent multiple options formatted precisely as you have
defined, but rather one option that has been split along any
arbitrary point into multiple containers. When documenting an
example, then, try to make sure that the division point you select as
an example does not lie on a clean division of your option contents -
place it at an offset so as to reinforce that these values must be
concatenated rather than processed individually.

DHCPv4 option fragments are a basic protocol feature, so there
usually is no reason to mention this feature in new option
definitions, unless of course the option is very likely to exceed 255
bytes, or the documented example(s) are this big.

Note that option fragmentation is also a very common side-effect of
running out of options space, and moving to overloaded FILE or SNAME
fields. Although the option may be considerably shorter than 255
bytes, if it does not fit in the remaining space then software may
consume all remaining options space with one option fragment, and
place the remainder in an overloaded field.

9. Clients Request their Options

In both DHCP protocols, there exists as part of the protocol
definition an option whose purpose is twofold - to inform the server
what option(s) the client supports and is willing to digest, and in
what order of priority the client places those option contents (in
the event that they will not fit in the packet, later options are to
be dropped).

It doesn't make sense for some options to appear on this parameter
request list, such as those are formed by elements of the protocol's
internal workings, or are formed on either end by DHCP-level software
engaged in some exchange of information. When in any form of doubt,
assume that any new option must be present on the relevant option
request list if the client desires it.

It is a frequent mistake of option draft authors, then, to create
text that implies that a server will simply provide the new option,
and clients will digest it. Generally, it's best to also specify
that clients MUST place the new option code on the relevant list

   option, clients MAY include the new option in their packets to
   servers with hints as to values they desire, and servers MAY respond
   with the option contents if they have been so configured.

[10]. Security Considerations

   DHCP does have an Authentication mechanism ([24], [2], [25]), where
   it is possible for DHCP software to discriminate between authentic
   endpoints and men in the middle.

   However, at this date the mechanism is poorly deployed. It also does
   not provide end-to-end encryption.

   So, while creating a new option, bear in mind that DHCP packet
   contents are always transmitted in the clear, and actual production
   use of the software will probably be vulnerable at least to men in
   the middle attacks from within the network, even where the network
   itself is protected from external attacks by firewalls.

   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 by defining a reasonable upper limit, or
   explicitly reminding the implementor that an option may be
   exceptionally long.

   For some option contents, "insane values" may be used to breach
   security. An IP address field might be made to carry a loopback
   address, or local broadcast 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 trusted,
   rather than validated.

   So it behooves an option's definition to contain any validation
   measures as can reasonably be made.

11. IANA Considerations

   This document has no actions for IANA.

Appendix A. Background on ISC DHCP

   The ISC DHCP software package was mostly written by Ted Lemon in
   cooperation with Nominum, Inc. Since then, it has been given to
   Internet Systems Consortium, Inc. ("ISC") where it has been
   maintained in the public interest by contributors and ISC employees.

   It includes a DHCP Server, Relay, and Client implementation, with a
   common library of DHCP protocol handling procedures.

   The DHCP Client may be found on some Linux distributions, and FreeBSD
   5 and earlier. Variations ("Forks") of older versions of the client
   may be found on several BSDs, including FreeBSD 6 and later.

   The DHCP Server implementation is known to be in wide use by many
   Unix-based servers, and comes pre-installed on most Linux
   distributions.

   The ISC DHCP Software Suite has to allow:

   o Administrators to configure arbitrary DHCP Option Wire Formats for
      options that either were not published at the time the software
      released, or are of the System Administrator's invention (such as
      'Site-Local' [26] options), or finally were of Vendor design
      (Vendor Encapsulated Options [5] or similar).

   o Pre-defined names and formats of options allocated by IANA and
      defined by the IETF Standards body.

   o Applications deriving their configuration parameters from values
      provided by these options to receive and understand their content.
      Often, the binary format on the wire is not helpful or digestable
      by, for example, 'ifconfig' or '/etc/resolv.conf'.

   So, one can imagine that this would require a number of software
   functions:

   1. To read operator-written configuration value into memory.

   2. To write the in-memory representation into protocol wire format.

   3. To read the protocol wire format into memory.

   4. To write the in-memory format to persistent storage (to cache
       across reboots for example).

   5. To write the in-memory format to a format that can be consumed by
       applications.

   If every option were formatted differently and uniquely, then we
   would have to write 6 functions for every option. As there is the
   potential for as many as 254 DHCPv4 options, or 65536 DHCPv6 options,
   not to mention the various encapsulated spaces ("suboptions"), this
   is not scalable.

   One simple trick is to make the in-memory format the same as the wire
   format. This reduces the number of functions required from 5 to 4.
   This is not always workable - sometimes an intermediary format is
   required, but it is a good general case.

   Another simple trick is to use the same (or very nearly the same)
   format for persistent storage as is used to convey parameters to
   applications. This reduces the number of functions again from 4 to
   3.

   This is still an intractable number of functions per each DHCP
   option. So, we need a way to reduce this to small orders.

Appendix A.1. Atomic DHCP

   To accomplish these goals, a common "Format String" is used to
   describe, in abstract, all of the above. Each byte in this format
   string represents a "DHCP Atom". We chain these 'atoms' together,
   forming a sort of molecular structure for a particular DHCP Option.

   Configuration Syntax language allows the user to construct such a
   format string without having to understand how the Atom is encoded on
   the wire, and how it is configured or presented.

   You can reasonably imagine that the various common formats of DHCP
   options described above (Table 1) each have an Atom associated with
   it. There are special use Atoms, such as one to repeat the previous
   Atoms indefinitely (for example, for options with multiple IPv4
   addresses one after the other), and one which makes the previous Atom
   optional.

   As the software encounters a format string, it processes each Atom
   individually to read, formulate in memory, or write to output the
   various option contents.

   The format strings themselves are either hard coded by the software

in a table of option definitions, or are compiled at runtime through
configuration syntax generated by the operator.

        option [space].[option] code [number] = [definition];

The "space" refers to the option space, which may be the DHCPv4
option space, the DHCPv6 option space, or any suboption space such as
the DHCPv4 Relay Agent Information suboptions or similar.

The "option" refers to the option's symbolic name within that space.

The code number refers to the binary code assigned to this option.

The definition is a complex statement that brings together DHCP
Atoms, many of which are the aforementioned common formats, that
compose this option. For example, here are two predefined options,
as they might have been configured for use by an operator if the
software did not already support them.

    option dhcp.path-mtu-plateau-table code 25 =
                                    array of unsigned integer 16;
    option dhcp.static-routes code 33 = array of { ip-address,
                                            ip-address };

    option dhcp.path-mtu-plataeu-table 4352, 1500, 576;
    option dhcp.static-routes 10.10.10.10 10.10.10.9,
                        10.10.10.11 10.10.10.9;

12. Informative References

[1]   Droms, R., "Dynamic Host Configuration Protocol", RFC 2131,
      March 1997.

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

[3]   Stapp, M., Volz, B., and Y. Rekhter, "The Dynamic Host
      Configuration Protocol (DHCP) Client Fully Qualified Domain
      Name (FQDN) Option", RFC 4702, October 2006.

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

[5]   Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor
      Extensions", RFC 2132, March 1997.

   [6]   Waters, G., "The IPv4 Subnet Selection Option for DHCP",
          RFC 3011, November 2000.

   [7]   Kinnear, K., Stapp, M., Johnson, R., and J. Kumarasamy, "Link
          Selection sub-option for the Relay Agent Information Option for
          DHCPv4", RFC 3527, April 2003.

   [8]   Provan, D., "DHCP Options for Novell Directory Services",
          RFC 2241, November 1997.

   [9]   Droms, R. and K. Fong, "NetWare/IP Domain Name and
          Information", RFC 2242, November 1997.

   [10]   Beser, B. and P. Duffy, "Dynamic Host Configuration Protocol
          (DHCP) Option for CableLabs Client Configuration", RFC 3495,
          March 2003.

   [11]   Luehrs, K., Woundy, R., Bevilacqua, J., and N. Davoust, "Key
          Distribution Center (KDC) Server Address Sub-option for the
          Dynamic Host Configuration Protocol (DHCP) CableLabs Client
          Configuration (CCC) Option", RFC 3634, December 2003.

   [12]   Monia, C., Tseng, J., and K. Gibbons, "The IPv4 Dynamic Host
          Configuration Protocol (DHCP) Option for the Internet Storage
          Name Service", RFC 4174, September 2005.

   [13]   Lemon, T., Cheshire, S., and B. Volz, "The Classless Static
          Route Option for Dynamic Host Configuration Protocol (DHCP)
          version 4", RFC 3442, December 2002.

   [14]   Schulzrinne, H. and B. Volz, "Dynamic Host Configuration
          Protocol (DHCPv6) Options for Session Initiation Protocol (SIP)
          Servers", RFC 3319, July 2003.

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

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

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

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

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

   [20]   Mockapetris, P., "Domain names - implementation and
          specification", STD 13, RFC 1035, November 1987.

   [21]   Aboba, B. and S. Cheshire, "Dynamic Host Configuration Protocol
          (DHCP) Domain Search Option", RFC 3397, November 2002.

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

   [23]   Littlefield, J., "Vendor-Identifying Vendor Options for Dynamic
          Host Configuration Protocol version 4 (DHCPv4)", RFC 3925,
          October 2004.

   [24]   Droms, R. and W. Arbaugh, "Authentication for DHCP Messages",
          RFC 3118, June 2001.

   [25]   Stapp, M. and T. Lemon, "The Authentication Suboption for the
          Dynamic Host Configuration Protocol (DHCP) Relay Agent Option",
          RFC 4030, March 2005.

   [26]   Volz, B., "Reclassifying Dynamic Host Configuration Protocol
          version 4 (DHCPv4) Options", RFC 3942, November 2004.

Author's Address

   David W. Hankins
   Internet Systems Consortium, Inc.
   950 Charter Street
   Redwood City, CA
   US

   Phone: +1 650 423 1307
   Email: David_Hankins@isc.org

Full Copyright Statement

Intellectual Property

Acknowledgment