Dynamic Host Configuration Working Group
Internet-Draft
Intended status: Informational
Expires: December 21, 2012

D. Hankins
Google
T. Mrugalski
M. Siodelski
ISC
S. Jiang
Huawei Technologies Co., Ltd
S. Krishnan
Ericsson
June 19, 2012

Guidelines for Creating New DHCPv6 Options
draft-ietf-dhc-option-guidelines-08

Abstract

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

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 upto 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 requries new 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.

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 network must also provide (such as domain name servers, or routers). This means that in addition to the existence of a configuration parameter, one must also ask themselves if it is reasonable for this parameter to be set by the directly attached network's administrators.

Note that the client still reserves the right to ignore values received via DHCPv6 (for example, due to having a value manually configured by its own operator). Bear in mind that doing so might cause the client to be rejected network attachment privileges, and this is one main reason for the use of DHCPv6 in corporate enterprises.

4. General Principles

The primary guiding principle to follow in order to enhance an option's adoptability is simplification. More specifically, 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: A bulk class of 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

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 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 value at the tail end of the option.

This estimates that implementations will 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 preferrable 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 fragments 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.

5.1. Option with IPv6 addresses

This option format is used to carry one or many IPv6 addresses:

```
      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 [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]

5.2. 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 2: Option with 32-bit-integer value

Examples of use:

o Information Refresh Time [RFC4242]

5.3. 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 3: Option with 16-bit integer value

Examples of use:

o Elapsed Time [RFC3315]

5.4. 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 4: Option with 8-bit integer value

Examples of use:

o DHCPv6 Preference [RFC3315]

5.5. 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. Some of the existing DHCPv6 options use NVT-ASCII strings to encode: filenames, host or domain names, protocol features or textual messages such as verbose error indicators.

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 5: Option with variale length data

Examples of use:

o Client Identifier [RFC3315]

o Server Identifier [RFC3315]

o Boot File URL [RFC5970]

5.6. Option with DNS Wire Format Domain Name List

This option is used to carry 'domain search' lists or any host or domain name:

```
    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 6: Option with DNS Wire Format Domain Name List

Examples of use:

o SIP Servers Domain Name List [RFC3319]

o NIS Domain Name [RFC3898]

6. Avoid Conditional Formatting

Placing a 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. So conditional formatting requires new code to be written, as well as introduces an implementation problem; as it requires that all speakers implement all current and future conditional formats.

Conditional formatting is not recommended, except in cases where the DHCPv6 option has already been deployed experimentally, and all but one conditional format is deprecated.

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 a 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 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 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).

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.

8. Suboptions 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 sub-options. It should be noted that, contrary to DHCPv4, there is no shortage of option numbers. Therefore 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.

Such encapsulation mechanism 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 sub-option inside IAPREFIX (PD_EXCLUDE, defined in [RFC6603]). It seems awkward to refer to such options as sub-sub-option, therefore "sub-option" term is typically used, regardless of the nesting level.

When defining configuration means for more complex mechanisms, it may be tempting to simply use sub-options. That should usually be avoided, as it increases complexity of the parser. It is much easier, faster and less error prone to parse larger 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 sub-options.

9. Additional States Considered Harmful

DHCP is a protocol designed for provisioning nodes. Less experienced protocol designers often assume that it is easy to define an option that will convey a different parameter for each node in a network. Such problems arose during designs of MAP [I-D.mdt-softwire-map-dhcp-option] and 4rd [I-D.ietf-softwire-4rd]. While it would be easier for provisioned nodes to get ready to use per node option values, such requirement puts exceedingly large loads on the server side. Alternatives should be considered, if possible. As an example, [I-D.mdt-softwire-map-dhcp-option] was designed in a way that all nodes are provisioned with the same set of MAP options and each provisioned node uses its unique address and delegated prefix to generate node-specific information. Such 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. For renewing other parameters, please use Information Refresh Time Option (defined in [RFC4242]). Introducing additional timers make deployment unnecessarily complex. Please try to avoid it.

10. Is DHCPv6 dynamic?

DHCPv6 stands for Dynamic Host Configuration Protocol for IPv6. Contrary to its name, is many contexts it is not dynamic. While designing DHCPv6 options, it is worth noting that there is no reliable way to instantly notify clients that something has happened, e.g. parameter value has changed. There is a RECONFIGURE mechanism, but it has several serious drawbacks that makes its use difficult. First, its support is optional and many client implementations do not support it. To use reconfigure mechanism, server must use its secret nonce. That means that provisioning server is the only one that can initiate reconfiguration. Other servers do not know it and cannot trigger reconfiguration. Therefore the only reliable way for clients to refresh their configuration is to wait till T1 expires.

11. Multiple provisioning domains

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

DHCPv6 was not initially designed with multiple provisioning domains. Although [RFC3315] states that a client that receives more than one ADVERTISE message, may respond to one or more, such capability was never observed in any known implementations. Existing clients will pick one server and will continue configuration process with that server, ignoring all other servers.

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.

12. Considerations for Creating New Formats

If the option simply will not fit into any existing work by using fragments, the last recourse is to create a new format to fit.

When doing so, it is not enough to gauge whether 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's fragments might reasonably have any other uses, and if so, to create the option with the foreknowledge that its parts may later become a common fragment.

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.

13. Option Size

DHCPv6 [RFC3315] allows for packet sizes up to 64KB. First, through its use of link-local addresses, it steps aside 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 impossible to describe any fixed limit that cleanly divides those too big from the workable.

It is advantageous to prefer option formats which contain the desired information in the smallest form factor that satisfies the requirements.

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.

14. 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 is willing to consume.

It doesn't make sense for some options to appear on this Option Request Option, such as those formed by elements of the protocol's internal workings, or are formed on either end by DHCPv6-level software engaged in some exchange of information. When in doubt, 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 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).

Creators of DHCPv6 options MUST NOT require special ordering of options either in the relevant request option, or in the order of options 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 be noted that any requirement regarding option ordering will break down most existing implementations, as "order is not important" was one of the design priciples of DHCPv6 and many implementations follow it. For example, there are existing implementations that use hash maps for storing options, so forcing any particular order is not feasible without great deal of work. If options must be processed in any specific order (e.g. due to inter-dependency), use of option encapsulation should be considered.

15. Security Considerations

DHCPv6 does have an Authentication mechanism ([RFC3315]) that makes

it possible for DHCPv6 software to discriminate between authentic endpoints and men in the middle. Other authentication mechanisms may optionally be deployed. For example, the Secure DHCPv6 [I-D.ietf-dhc-secure-dhcpv6], based on Cryptographically Generated Addresses (CGA) [RFC3972], can provide source address ownership validation, message origin authentication and message integrity without requiring symmetric key pairs or supporting from any key management system. However, as of now, the mechanism is not widely deployed. It also does not provide end-to-end encryption.

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 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 implicitly trusted without being validated.

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

16. IANA Considerations

This document has no actions for IANA.

17. References

17.1. Informative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

[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.

[RFC3319] Schulzrinne, H. and B. Volz, "Dynamic Host Configuration Protocol (DHCPv6) Options for Session Initiation Protocol (SIP) Servers", RFC 3319, July 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.

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 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.

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

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

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

17.2. Informative References

[I-D.ietf-dhc-secure-dhcpv6] Jiang, S. and S. Shen, "Secure DHCPv6 Using CGAs", draft-ietf-dhc-secure-dhcpv6-06 (work in progress), March 2012.

[I-D.ietf-softwire-4rd] Despres, R., Penno, R., Lee, Y., Chen, G., and S. Jiang, "IPv4 Residual Deployment via IPv6 - a unified Stateless Solution (4rd)", draft-ietf-softwire-4rd-00 (work in progress), May 2012.

[I-D.mdt-softwire-map-dhcp-option] Mrugalski, T., Boucadair, M., Deng, X., Troan, O., and C. Bao, "DHCPv6 Options for Mapping of Address and Port", draft-mdt-softwire-map-dhcp-option-02 (work in progress), January 2012.

Authors' Addresses

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

Email: dhankins@google.com

Tomasz 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

Phone:
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