Network Working Group
Internet-Draft
Category: Informational
Oct 1999

R. B. Hibbs, Pacific*Bell
N. Lane, Wal-Mart Stores

Interpreting Client Options for the Dynamic Host Configuration Protocol

<draft-ietf-dhc-client-options-00.txt>

Saved: Thursday, October 14, 1999, 2:24 PM

Status of this Memo

Copyright Notice

Abstract

During the summer of 1999, a grand debate raged over the correct interpretation of several DHCP client options as described in [RFC 2132], as well as the need for one option whose proposing

Internet-Draft expired.

As a result of that debate, the authors gained some insights into the intended (or unintended!) interpretation of certain options defined in [RFC 2132,] particularly the Vendor Class Identifier (option 60) and Vendor Encapsulated Options (option 43.)

These insights are presented in this informational Internet-Draft, whose reason for being is to act as an aid to implementers of the DHC protocol, and to future editors of the underlying RFCs and selected, current Internet-Drafts. This memo is not being proposed as a standards-track document, but rather as an aid to clarify existing and future RFCs.

Network Working Group R. B. Hibbs, Pacific*Bell
Internet-Draft N. Lane, Wal-Mart Stores
Category: Informational Oct 1999

Table of Contents

1. Introduction

This memo was produced by the DHCP Working Group and attempts to identify and clarify a few specific cases where the use of client options is not rigorously specified by the Dynamic Host Configuration Protocol.

This memo does not cover every DHCP/BOOTP client option nor every element of a DHCP/BOOTP request/response packet.

This memo is based on the Internet standards-track DHC protocol as defined by documents [RFC2131 and RFC2132].

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 document [RFC2119].

2. Overview

DHCP is widely used by many different vendors of computer and networking hardware and software to provide a straightforward means of supplying IP address and networking configuration data to individual client hosts from DHCP servers. The Requests for Comments (RFCs) that specify the protocol and the configuration elements that may be specified by a DHCP message exchange have grown significantly as the protocol has become more widely deployed until we find ourselves in the situation we experience today with nearly 100 options the network administrator can use to perform semi-automatic configuration of client hosts. The proliferation of options has led to a small number of cases where the interaction among options is not rigorously specified, causing confusion or interoperability failures. This RFC attempts to identify some of these cases and clarify the expected behavior of both client and server.

In this memo, the specific cases to be studied will be first identified and, hopefully, clarified, then some of the discussion that lead to the author's contentions about the correct use of the client options will be presented to show the rationale for inclusion.

At the time of writing, the authors do not know the eventual form of the investigation that led to the production of this memo. Three alternatives exist: (1) publication as an "informational" RFC, (2) publication as a "best computing practices" (BCP) memo, or (3) justification for revision of the basic DHCP RFCs. None of these is preferred by the authors over any other. Presumably the DHC Working Group will review and decide the best course for the memo.

3. Cases

3.1. Vendor Classing

Vendor classing is provided through the use of options 60 (Vendor Class Identifier) and 43 (Vendor Encapsulated Options), in conjunction with option 55 (Parameter Request List.)

3.1.1. Classification Scheme

Vendor classing attempts to address the question of "How do I classify a client such that the client receives appropriate configuration data for their specific situation?" While every deployment of DHCP will have its own unique characteristics, consider a large organization with geographically-dispersed locations where clients requiring DHC services may be in different organizational entities, with different user processing functions, and of different generations and types. It is likely that the client population can be viewed a number of different ways, such as:

1. Geographical ("Where is the client located?")

2. Organizational ("In which department is the client

located?")

3. Functional ("What job does the user perform?")

4. Regulatory ("What statutory constraints affect the user?")

5. Networking ("How is the client connected to other hosts?")

6. Environmental ("What software is the client using?")

7. Platform ("What hardware is the client using?")

Some organizations may have fewer, others more, than these different views. Each view, especially the latter three given above, may be further subdivided: for example, Environment might have as many as four dimensions (operating system, TCP/IP network stack, DHCP client, and principal application software) while Platform most likely always has two (system manufacturer's model, network interface vendor's model.) RFCs 2131 and 2132 do not specifically address how many views a network administrator could or should take of the environment under their purview, but the underlying intent seems to be that all necessary and sufficient information to permit informed configuration of client hosts ought to be available for exchange and use by the clients and servers.

Where does the Vendor Class fit in the hierarchy of client views? While it could legitimately be argued that any "vendor-supplied" component of the client (either hardware or software) is a candidate, one of the editors believes the proper fit is aligned with either the TCP/IP stack or DHCP client based on the following argument:

Network interfaces numbered in conformance with IEEE standards contain a manufacturer's code as part of the interface's hardware address, which is already carried in the 'chaddr' field of a BOOTP packet. Assuming that Mike Henry's proposal for a Globally Unique Identifier tied to specific host systems is accepted by systems manufacturers there will be a way to completely identify [newer] systems unambiguously. Having covered the two primary views of the hardware platform, the remaining "vendor-supplied" components are software (or, possibly, embedded firmware such as a writeable control store or flash PROM.) It will be argued in the next section that application software is a user, not vendor, characteristic. As the underlying operating system software is much less important for determining client networking behavior than the choice of the TCP/IP stack or DHCP client, the editors propose discounting operating system as a factor.

In some cases LAA devices, where the vendor's MAC address is replaced with a Locally-Assigned Address from the range 400000 (assigned by the IEEE for local addressing), it may be necessary to override the Globally Unique Identifier that Mike Henry is proposing. In some environments LAAs are a requirement for in order to effectively manage BOOTP devices. However, when talking DHCP, we should REQUIRE the use of the Client Identifier in lieu of the LAA, so the conflict may not be as great. (Should we also require that the Client Identifier be configurable by an administrator?)

3.1.2. Mode of Operation

The basic mode of operation for vendor classing is that during the discovery phase the client broadcasts a DHCPDISCOVER message containing Vendor Class Information (option 60) which the server optionally uses to select an IP address and other configuration information to offer to the client in a DHCPOFFER message. Note the word "optionally." RFC2132 stops short of mandating any specific kind of server behavior on receipt of this option. This is not quite an oversight by the RFC editor: the editor had to consider the possibility that any specific server might not be configured to recognize a particular Vendor Class Identifier. As that case is an implementation issue under control of the network administrator, not the editor, any server response MUST be optional.

Let's continue by assuming that the server has been configured to recognize the Vendor Class Identifier sent by the client, and has stored some specific data to be used to configure any client belonging to that class. What does the server do? There are essentially three approaches: use the Vendor Encapsulated Options string (option 43) to return data to the client, return data in one of the options specified in the "Host Requirements" RFC [must supply RFC number here û ed.], or return data in some other option.

Which approach is correct? Actually, all of them are! The one chosen by a specific client-server pairing is a matter that SHOULD be specified by the vendor who declares the need for vendor-specific options data. While no hard and fast rules apply, generally, if the data to be returned is covered by the Host Requirements, it SHOULD be returned in the proper option. If not covered by Host Requirements, but is covered by another, existing option, that option should be used. The Vendor Encapsulated Options string should be used for data that fits within the limitations of a BOOTP/DHCP option field (0-255 octets) that doesn't also fit any other existing option. Note that except for options necessitated by the Host Requirements, the option number of the correct option(s) MUST be included in the Parameter Request List (option 55) or the server has no obligation to return it to the client.

What about longer aggregations of configuration data than will fit in a single option? The editors are not prepared to offer a general solution to this problem but will suggest that it may be possible to use existing protocol facilities, such as 'file' or 'sname' to accomplish transfer of larger amounts of configuration data. Clearly, this is an area for more study.

3.2. User Classing

User classing is provided through the use of option NN (User Class Identifier) in conjunction with option 55. While it basically performs a similar function to vendor classing, it differs in one major respect: there is no User Encapsulated Options data specified for DHCP.

The history of User Classing is a bit murky. First proposed (if our memory is correct!) by Glenn Stump, the Internet-Draft of this option was allowed to expire, then it was resurrected to the objections of some members of the Working Group, and has again fallen into limbo. The editors believe it continues to have value and should be reinstated. An example will hopefully illustrate:

Suppose your organization operates a large call center, large enough to warrant its own tandem switch which contains an adjunct processor that includes DNS service for client workstations matching client telephone sets. Further, suppose that in order to perform database lookup of customer data based on incoming Automatic Number Identification data and answer incoming calls with 400 milliseconds, there is an insufficient time budget to perform certain functions such as dynamic DNS update or even dynamic assignment of IP addresses for newly-activated clients. This entire group of clients are a natural grouping whose user characteristics differ considerably from all others within your organization. At the very minimum, they should be associated with the adjunct DNS server rather than your organization's primary DNS servers. Perhaps they are to be given a unique Domain Name.

The editors are neutral as to whether or not the User Class Identifier option should have a corresponding User Encapsulated Options String option, similar to option 43, but do believe that the User Class Identifier should be part of the DHCP options universe.

3.3 Client Identifiers

The DHCP Client Identifier (option 63) is specified in RFC2131 as the primary key for locating IP address leases by both client and server, yet considerable misunderstanding remains about this option. Specific issues concern the uniqueness of the Client Identifier and how the uniqueness influences selection of an IP address lease to offer a client.

An early design decision for DHCP was to require the Client Identifier to be unique only within a network segment. This design choice permits roaming by mobile clients among a group of disjoint subnets, and is a major convenience for implementers of DHCP. Enlarging the scope of uniqueness for the Client Identifier would "break" many existing installations, so is considered to be "out-of-bounds" for future discussion.

As mentioned in section 3.1, Mike Henry of Intel Corporation proposed a Globally Unique Identifier to the DHC Working group for those instances where a unique identifier with scope greater than a network segment is required.

Also, as mentioned in section 3.1.1, the editors believe that providing an administrator the ability to configure the Client Identifier may be a desirable feature for clients, especially if the Globally Unique Identifier (tied to non-accessible hardware identification) is available, and User Classing is not. This would permit finer control of DHCP-supplied client configurations by permitting more precise identification of the group to which a particular client belongs.

3.4 Option Default Values

What happens when either the client does not request an option essential to its operation, or the server is not configured to provide that data (through oversight or administrative error)?

Are there reasonable default values that can be recommended for specific options? While some values may be suggested [or required] by Host Requirements and other RFCs, the editors believe than a generally-accepted set of defaults that may be assumed by a client or server deserves some additional study.

3.4.1 IP Stack Options

The editors know of very few DHCP clients that actually request "path-mtu-discovery", but the host requirements RFCs state a "reasonable" default for this value. Here there is a conflict between the virtues of brevity, that is, supplying clients with the smallest set of options necessary to begin functioning as a host on the network, and the desirability of requiring minimal assumed values by clients. The editors have encountered several clients that were badly implemented, calling for every option (all 254!) in a Parameter Request List because they made no attempt to understand or resolve the Host Requirements.

A related question is whether a DHCP server SHOULD send options it knows to be required for successful operation (e.g., Router Address) even if the client does not request them? The editors know that many servers do send a minimal list of options, but is there any need for agreement as to what constitutes a minimum set?

3.4.2 Other Options

Aside for options whose presence is required by various RFCs (e.g., DHCP Message Type) is there any need to identify a minimal set of other options to provide a client? Is there any need to codify default values for options not mandated by the RFCs?

3.5 Who Wins in a Conflict?

In many of the Internet protocols, there are well-established rules for settling conflicts that may arise in operation, for example, the "lowest bid wins" rule often applies for durations or lifetimes. When considering DHCP client options, can a consistent and defensible set of guidelines or rules be established to determine whether the client or server "wins" when a conflict arises?

Taking the example of lease duration, the server is required only to offer a lease to a client that is of satisfactory duration to the serverù if the client wants a longer lease, it merely chooses not to request assignment of the offered lease. The assumption is that the client can choose among competing offers, selecting the one it prefers. Is there any need to recommend client behavior if it should not like any of the offered leases.

The editors believe it is important to recommend client behavior upon non-acceptance of an IP lease. Some sites have defined this behavior for their clients, especially in reference to IPv4 autoconfiguration (not-enforceable until the clients actually honor the new DHCP "do not autoconfigure" option), but is a general policy appropriate?

The editors suggest the following client behavior: If a client is offered a lease without acceptable parameters, but still allows IP-level connectivity to function, the client MUST accept the. If the client does not get a lease or if it is told not to configure the IP stack, the client MAY continue trying at the standard exponential backoff intervals as specified in RFC 2131.

For many administrators it is crucial that clients accept valid leases offered to them: Failure to do so may result in a non-functioning client. The editors are undecided if it is a good or poor idea to permit clients to suggest values for options such as the DNS Server, but we are in agreement that if a client were to refuse all offered leases because a critical parameter didn't satisfy the client's notion of what it was willing to accept, then chaos would reign in the network.

4. Discussion

4.1 Vendor Classing

The following discussion took place on the ISC dhcp-server mailing list during June, July and August of 1999, and has been edited considerably to preserve the context of each comment while eliminating unnecessary remarks:

[Bahman Sistany]

I have been looking for any "published" vendor-specific options by vendors and I cannot find any. Does anyone know of a situation where these options are being used (i.e., requested by a DHCP client and their values returned by a DHCP server?)

The ISC dhcpd ignores any unrecognized options such as Vendor Specific Options as the protocol suggests it can do. But if the server were to return values for these options it would need to know about them in advance. That's what I mean by "published" above.

[Mike Henry]

The PXE specification from Intel addresses this area. At http://developer.intel.com/ial/wfm/tools/index.htm you can find specifications (including a generally useful set of vendor-specific options), source code and binaries for the NT and Linux environments to provide proxies for DHCP services not capable of parsing Option 60, and the same for a boot service (daemon) to allow a heterogeneous set of bootservers for the booting client to choose from.

[unknown]

Sun JavaStations, the Solaris 7 boot system (I think), the Solaris 8 boot system (definitely), Sun workstation network boot ROMS.... You're probably seeing a pattern here. I think vendors other than Sun are using vendor encapsulated options, but I don't know of any off the top of my head.

[Nathan Lane]

Unfortunately, there is a great lack of understanding in the industry what these options are for. I know of one thin client vendor that packs in the parameter request list ALL 127 site-specific options (128 through 254) then parses each one to see if their magic cookie is in the data of each option. Just off the top of my head, it looks to me like a vendor should send in the parameter request list option 60, vendor class identifier and then the server should respond with the information in option 43 (data which is totally opaque to the server.) Is this everyone else's understanding of how it should operate?

[Ted Lemon]

It's mine, anyway.

[Stuart Stevens]

It is my understanding that Windows 2000 will support 3 vendor options (1-3) and the Windows 2000 DHCP server is already programmed for these vendor options. I am not aware of these options being published.

I thought that option 43 is more appropriate for the parameter request list.

[Ted Lemon]

The client should send option 60 and request option 43 in the parameter request list.

[unknown]

The ISC DHCP server needs to know about them before the first client requests one, but since you can define them in the configuration file, it's not the case that the server needs to have them compiled in or anything like that.

[Nathan Lane]

[Client] behavior, at least to me, seems well-defined in RFCs 2131, 2132 and 1497. Our "site specific" option space, 128 through 254, is being polluted by vendors who don't understand or won't use vendor specific codes.

[Barr Hibbs]

I've seen the "request all" behavior from both Linux and thin clients, and I wonder why they are so clueless...

I'm not sure how the association between vendor class identifier (60) and vendor encapsulated options (43) can be inferred -- I see lots of clients send me option 43, but they almost never request it!

I've also had thin client vendors tell me that I should "capture" the data sent in option 43 and return it to them when requested!

This may be an area where the RFCs need to be reworded.

[Dave Gotwisner]

The only problem with using option 43 (or options 128-254) for the vendor specific options, as I understand it, is that option 43 is of the same format as the rest of the option space (i.e., a series of option numbers, followed by lengths and data) and there is no definition that states what vendors use what codes. If the DHCP server is not smart enough to send different option 43's (or any other option) based upon a vendor or client ID (note, this would typically need to be a wild-carded selection, since otherwise you hit the same problems of the old bootptab format -- proliferation of entries because of MAC values). ISC MAY be smart enough to be configured this way. Microsoft's DHCP server is not, at least without having to do a lot of work to get around their UI.

As someone else said, Microsoft 2000 is using vendor options 1, 2, and 3. Assuming this is correct, how do you deal with someone else who is also defining their device to look for these options (for completely different purposes), especially when both are deployed in the corporate environment?

RFC 2132 says that options 128 to 254 are reserved for site specific options. Vendors can read this two ways. I read it one way, Nathan, another. Is the site specificity specifically for the end user to use, or is it for manufacturers to provide optional capabilities (outside of RFC 2132) which a site may use if they want those capabilities? If I want to guarantee that I don't step on another vendor's custom option, the only way I can do it is to submit a set of options (via RFC) and go through the review process. This may be the best way, but the under option 128 space is filling up, and I don't think it is appropriate for each vendor to reserve blocks of a very limited space for their stuff (just look at some of the options already reserved (10, 14, 16, 68, 75, and 76 all come to mind)). Novell submitted their own set of options in RFC 2241 using 85 and 86.

We are a vendor. We make network terminals and Windows-based terminals. We have several different product lines, all of which want a different set of data for configuration. Our customer base has stated that they want to use DHCP to be able to configure the devices for ease of use. Many have also demanded plug and play capability -- you connect the device to your network and turn it on, everything that you need for running the device the way the customer wants autoconfigures itself from the network.

Some of our products hard code the set of options (in 128-254) that they use, and once the terminal comes up, you can configure it. If our choice of defaults is acceptable (i.e., doesn't conflict with other devices in the customer's environment) no other configuration need be done with respect to the DHCP set of options.

Our other products use string tags (TAG = value) with a custom tag prefix, which we put anywhere in the 128-254 option space. This allows the administrator to chose what options he/she wants to support and to guarantee that there isn't a conflict with other devices in the environment from the DHCP server side. Unfortunately, this approach forced us to request (via option 43) all options in 128-254.

Neither is a good solution. Unfortunately, there is no way to really protect against two devices (from two different vendors) from wanting the same option number for different purposes, whether they are through option 60 (in encapsulated form) or in 128-254.

DHCP servers which are smart enough (or configurable enough) to provide different data based upon Vendor ID, Client ID, or some other tag would do a great deal to reduce the problem, but not all DHCP server vendors do this.

Of course, servers which only support the minimum record size and fail to support Option Overload further exacerbate the problem.

Also, clients which support option 18 (Extensions Path) would also help, especially if the format of this option lent itself to editing the file. Unfortunately, many clients fail to support it without going outside the client.

[David Corlette]

Excellent points all, but it seems to me that there are lots of ways around the problems stated. For instance, why not pick a single option, register it so no other vendor "steps" on it, and then have that option contain the address of a server that can distribute configuration information for your terminals? As I understand it, this is similar to the TFTP server option; indeed, you could probably use that one if need be.

As for the configuration server, you could quite quickly write one up, as all it really needs to do is be a FTP or TFTP server, maybe with some custom code to detect what type of terminal is requesting.   Release the server as source code; it can run on the same machine as the DHCP server, and you could provide precompiled binaries for the major platforms.

This is how many other terminals and machines autoconfigure, why not yours? I'm sure there are other ways to deal with the issue, all of which avoid the overuse/misuse of the ill-defined vendor option codes. I mention the above only as an example of how simple it would be to limit your use of option codes to a single, already defined one.

[Brian Murrell]

You have to tell the server [which clients] are what kind [of devices]. This is how Merit RADIUS deals with the same problem in the "vendor-defined attribute" space. You tell it that a device is (say) a Livingston Portmaster and it uses the Livingston attribute space. You tell it that a device is an Ascend TNT and it uses the Ascend attribute space. This is quite manageable for things like NAS. Doing the same in a DHCP environment does get a whole lot uglier, I will admit. Perhaps the use of Client Identifiers will help this situation out. Perhaps vendors should be identifying themselves in the Client Identifier and that can be keyed (via wild cards, perhaps) to a vendor option space.

How about clients that do the same thing [support only the minimum packet size]?

[Ted Lemon]

The option codes in [option 43] are reserved for the vendor's use. Every DHCP server of which I am aware, with one possible exception, is capable of returning different values based on the value in option 60 that the vendor sends.

[Dave Gotwisner]

If you can tailor option 43 based upon what option 60 sends, you

should also be able to tailor what other options get sent, since you may want to use a different Boot File (option 13), swap server (option 16), extensions path (option 18), maximum record size (option 57), etc. Likewise, you should be able to send different 128-254 based upon option 60. Option 43 is limited in that it can't span the standard space + sname + file space's, individual options can span them.

[Ted Lemon]

This isn't a practical limitation, as long as you negotiate for a large DHCP packet using the Maximum Message Size option.   You aren't hoping to stuff your complete terminal configuration into a DHCP packet, I hope!

[Dave Gotwisner]

My issue is that [one vendor] uses vendor option 1, [a second] uses vendor option 1, [a third] uses vendor option 1. Without a smart server capable of triggering based upon option 60 (or equivalent), a site that uses all three products may get incorrect behavior on two of them, maybe even disastrous behavior making the device unusable.

[Ted Lemon]

The protocol specifically states that option one in the vendor option data is different for every vendor.

[Barr Hibbs]

Some of the problems [we've experienced with client implementations include;]

1. Clients would send five or six options in the 128-254 range in discover packets, and if we didn't return those in an ack packet, they would immediately cycle back to discover, without issuing a release, effectively abandoning the lease. Of course, they were offered the same lease again, and so we cycled endlessly, never committing a lease to the client....

2. If we didn't return options 58 and 59 (T1 and T2 values) they dropped into INIT-REBOOT state every 60 seconds and sent a new request packet.

3. Clients expected the TFTP server option to contain the address of the Winterm server. Curious, but not really a problem, just a misinformed use of the option.

4. lients complained when we didn't return hostname, even though it wasn't in the parameter request list. [The vendor] never budged on this one, completely ignoring the RFCs, including host requirements.

5. [One vendor] insisted that RFC2131 wasn't compliant with RFC1541 and refused to acknowledge that 2131 superseded 1541.

[Nathan Lane]

[One vendor] wouldn't use the "swap-server" option because they felt it was reserved for use by Sun Microsystems. They wouldn't use the "rootpath" option because it didn't exactly specify the swap path file (wouldn't conventional use be to append rootpath with "client-name.swapfile" or something like what Sun used to do with that option?)

[Mike Henry]

Interesting -- the fuzziness in this area (option 60 and option 43) is surprising, but in retrospect it certainly explains a bit of our confusion about guidance received in this area that didn't seem to match our reading of the specification. But then, we weren't very confident of our reading in the first place because the spec did not seem to be completely clear.

My reading of the text for option 43 (see below) is that the client is expected to use option 43 to send information about itself to the DHCP server and the DHCP server is expected use option 43 to send configuration information to the client that is specific to the client's vendor class. In both cases, the information in option 43 is specific to the vendor class indicated in option 60. Is this interpretation incorrect? Are there DHCP servers that actually parse incoming option 43?

The general direction we have been given is to embed client-specific information in the Vendor Class Identifier (option 60). This clearly can be made to work, but embedding a number of attributes in option 60 leads to creation of ad hoc formats for what amount to encapsulated options. Setting aside for a moment current DHCP service implementation, it seems like it would be more logical to put this "subclass" information into option 43 and leave option 60 to define the general class. This seems to be what option 43 text is saying. Am I misinterpreting the option 43 text?

Finally, taking the option 60 and option 43 text together, my impression is that the DHCP service is supposed to know what to do with option 60, but option 43 is opaque, to be interpreted by vendor specific code. Does this imply that DHCP services should have some means of plugging in vendor specific code to interpret option 43 and, presumably, generate the option 43 response to the client? Also, it seems to imply that the DHCP service should hand off processing to the vendor supplied code if the DHCP service sees an option 60!

[Bahman Sistany]

As far as I can tell, you are almost right. Here's my correction(s). Someone else will correct me if I misunderstand Vendor specific options as well. The client wants a specific vendor's options, say, vendor1. Here is part of what he'll send the server:

```
option code, length, value

60           7       vendor1

55           1       43
```

So this means I am [requesting vendor-specific options] for vendor1. The server should have something like the following in its config file:

```
vendor1 data1

vendor2 data2

...
```

When the server receives the request, it will send data1 in encapsulated form using option 43:

```
option code, length,     value

43           len(data1)  data1
```

Note that like you said data1 is opaque as far as the server is concerned and the client who asked for [the option data] should be able to interpret [the option] on its own. The server doesn't really care about this part though.

Here's a question on something related: My understanding is that the client can ask for specific options using option 55 (parameter request). The server doesn't have to supply those though. Also the server can send arbitrary options (not requested by the client) and the client can pick and choose among them or not use any of them at all. Is this right?

[Mike Henry]

Thanks, but I am afraid you only addressed the well-known half of the question. There is no doubt the DHCP service returns information to the client in encapsulated options in Option 43. However, the part I am really interested in is whether real "live" DHCP services have the ability to make use of encapsulated options within Option 43 that the client sends to the DHCP service. RFC 2132 seems to clearly say this is expected use of Option 43, but I am not aware of a DHCP service actually capable of providing this functionality.

Here is the 2132 text:

> "This option is used by clients and servers to exchange vendor-specific information. The information is an opaque object of n octets, presumably interpreted by vendor-specific code on the clients and servers."

[Bahman Sistany]

You are right in interpreting the protocol (I reread it again and compared it to RADIUS which in a very limited sense is a similar protocol). However, I have not heard of any DHCP servers that would actually use vs. info sent by the client (they ignore it [as permitted by the RFC.]) If a server has to use this info, then like you said it would have to load some [vendor-specific] code based on the value of option 60.

[Barr Hibbs]

Am I correct that what Mike and Bahman are asserting is that a DHCP server should not only accept option 43 from a client but should also do "something" with the received data? Does that "something" specifically include, in your understanding, returning the encapsulated data, verbatim, to the client if the client requests that option 43 be returned by its inclusion in the parameter request list?

While that is certainly possible, I wonder if that is the "right thing" to do because of the Pandora's box that it opens: I've already had problems, as Nathan has, with thin-client vendors who think that a DHCP server should be an external data store for a client. If your view of a server's role is to receive, store, and replay encapsulated data using option 43, then I don't know how we could prevent similar interpretations of a server's role for most other options.

I also wonder how you might resolve potential configuration and usage problems: suppose a server is configured with a specific set of vendor-encapsulated options for a specific vendor-class identifier and the client sends additional or different encapsulated options to the server as part of the protocol exchange -- what does the server do?

Finally, I wonder what was intended by the RFC text that Mike quotes, as it does seem to imply that a server ought to be able to take some action based on the receipt of vendor encapsulated options.

[Kevin Bracey]

As I understand it, the logic looks like this: The client MAY send option 60, vendor class. If it does, it may also send option 43, vendor specific options.

If the server gets option 60, and recognizes the vendor class:

1. Process the vendor specific options in a vendor-specific way (specific to the vendor class of the client). This may affect what is returned. For example, you might have a vendor specific option for a device to specify its memory size, which might lead the server to return a different boot file.

2. Return any standard options suitable for that vendor class.

3. Put any vendor-specific options for the client in option 43.

Option 43 can mean anything the client wants, in either direction -- what it means depends on the vendor class. The suboptions could be totally different in the two directions, although that would probably be a bad design decision.

The behavior you describe of "storing" option 43 would be one permissible use of it, as "vendor-specific" allows anything. It would take more than a [server configuration] tweak to achieve that though, and it would have to be a particular behavior invoked only for certain known vendor classes.

[Barr Hibbs]

This gets to the heart of the issue that Mike Henry raised!

...just what does "Process the ... options ..." actually mean to you (and anyone else who wishes to comment)? If you are suggesting that an arbitrary server must somehow not only be configured to identify the vendor class but also perform some vendor-specific processing on the encapsulated options which may have been sent by the client, just how does the vendor/ user/ administrator "know" what processing to perform? Do you imagine that a vendor would develop "plug-ins" for popular DHCP servers? Would DHCP servers be required to publish an API to permit plug-ins? Do you expect that IETF would codify the interface in an RFC? This gets nasty "real quick now!"

...I have no problem at all with option 43 containing opaque values, which is the current state of RFC 2132, for the server to return from its configuration to a client when requested, but if you are suggesting that the server somehow accepts one set for some purpose then returns another set, I really would have to see a convincing argument to support that.... I think it would be an implementation nightmare with very, very few benefits to offset the monumental headaches.

...my specific objection to storing, then replaying, any option (not just 43) is that such behavior turns the DHCP server from an information provider to a limited file server or database system, neither of which is an appropriate use for a service which is intended only to provide the networking configuration for acceptable clients. My servers support over 118,000 clients: if I had to store up to 255 bytes of data for 254 options for all of my clients, then the additional storage requirement for my servers could be as great as 7.6 Gbytes! This is because I don't believe you could prevent nearly every option from being used this way: after all, why couldn't a client "suggest" which name servers or routers it prefers to use?

This is a great deal more than merely tweaking [the server configuration] -- it is, I believe, a complete change in the way some of us believe a DHCP service should operate. If a client really needs a file service to save data between reboots, then it should do so with some server intended to be a data repository, not try to piggy-back onto DHCP, which really is not intended for that purpose. I also can't imagine how any DHCP server could effectively implement per-vendor processing of options where the server actually manipulates what is supposed to be opaque data (option 43).

[Nathan Lane]

I can see this need [to process vendor-specific options in a vendor-specific way.] I think there must be a better or different way.

Yes, the DHCP server does indeed know how to process vendor class options on a vendor by vendor basis and will pick a vendor specific option 43 to send only to that vendor's product. It does NOT get as complicated as you mention, though, in my opinion. The server designer and implementer must actually communicate with the developer implementing the device and get the specifications and requirements from the vendor. No more can a device implementer envision that DHCP only provides an IP address. It is the host configuration protocol, so it reasons that one should use it to configure as much as possible in the device that relates to its network configuration and connectivity. It sure does make a server implementer's job much more complex, but I think it is a reasonable step to take. The device implementer's desires are for "plug and play network". I feel DHCP's job is to facilitate that idea.

I feel it is absolutely not the DHCP server's responsibility to actually touch or process any data within option 43 and, according to my interpretation of the RFCs, the client has no business even sending an option 43.

[It should] not be necessary [that a vendor develop "plug-ins" for popular DHCP servers.] A vendor class would become a fairly overloaded field, but I think it is appropriate in this example. I see the configuration like this (pseudo code):

```
     if [ vendor-class-identifier = "SUN-JAVASTATION-8MB" ]
     then send option-43

        "sun-specific-data-for-an-8MB-javastation"

     else

        if  ....

        endif

     endif

   # resume normal option processing to build the
   # rest of the response packet.
```

Yes, this would require most servers out there right now to be modified and a scripting like language possibly be established.

It is not the IETF's job to specify any DHCP vendor's API or interface [to a DHCP server to permit "plug-ins."] Perhaps the IETF should specify how a DHCP server should make the information available to an externalized process (I'm not using externalized to mean a callout; just generically) or script language. I believe the ISC server's direction is to make some kind of embedded language available for just this type of thing. It is nearing a requirement in my environment that servers do implement some kind of regular expression pattern matching on DHCP input packets and intelligently decide, via external policy lookups, what should be done with the device. It is much more than just "to give an IP or not to give an IP."

I really do not see option 43 as a bi-directional communications channel. It is one way only.

No, I don't support [storing, then replaying any option] either. A DHCP server is not a little 255 byte or so data store the client can use to stash that information. However, if I have a vendor class of "SUN-JAVASTATION," I DO want to be able to send a pre-configured option 43 to the client.

Again, no way. I [also] couldn't support a client suggesting what it wanted [for every option] -- that would be mighty presumptuous of it!

I don't see the server as actually manipulating the opaque data. I see the server intelligently choosing which set of opaque data to send to which set of vendor classes.

I strongly think it is time to clarify the clarifications in regards to vendor encapsulated options and their behavior. I think we should take this to DHC and start working up a draft. I have a good base for one that is an internal document I'm completing describing our internal requirements for a DHCP client.

[Nicolas Williams]

DHCP's original purpose was to allow clients to obtain a reasonably small set of configuration information needed to connect to a network and where the clients know nothing more than a simple Client Identifier which can be as simple as the vendor provided hardware address of the vendor provided network interface.

This "small set of configuration information" means network address and routing configuration + name service configuration + [optionally] boot file server and path. The client can go from there.

It would be best if DHCP continued to do nothing more than that.

If any software on the client needs to be configured beyond the above, then a different protocol should then be used to retrieve the information from a configuration server; this is much easier to do when a client has become a full-fledged node in a network and there's no excuse for a client not to be able to do this today via DHCP.

What's missing is an open protocol and data format standard for storing, administering and obtaining post-DHCP configuration information. Some platforms offer their own system to do this, usually based on a proprietary name service system; I'm thinking of NetInfo (for NextStep/OpenStep/MacOS), NIS/NIS+ (Sun et. al.), Active Directory (Microsoft, Cisco), even flat files (for poor administrators).

[It] is reasonable [not to expect the server as actually manipulating opaque data,] but there will always be people for whom it's not enough.

[Barr Hibbs]

The pseudo-code fragment from Nathan's note is a pretty concise statement of how I believe that options 60 and 43 should interact. Given that, I imagine that it is a vendor's responsibility to offer network administrators something like the following:

"PDQ tiny-stations (tm) identify themselves to a DHCP server by sending a vendor class identifier (option 60) that specifically names the tiny-station sending the request. Series 100's send the string 'pdq-tiny-station-100' while series 250's send the string 'pdq-tiny-station-250.' If your tiny-station series 100 contains the optional writeable control store (model 103) your DHCP server should return the hex value '30:cf:12:59:72:21' as a vendor encapsulated option 43...."

Then it would be the responsibility of the network administration to ensure that, like most other bits and pieces of configuration data, the precise vendor encapsulated option (if any) for a specific vendor class identifier is included in the server configuration.

I also agree that the vendor class identifier could be used in similar ways with other options, for example:

```
if [ vendor-class-identifier = "SONY Playstation SE"

then

send interface-mtu 768

endif
```

I believe that is a consistent use of the class identifier as well.

I think we are actually closing on the essential points here.

I'll have to defer to Ralph about original purposes, but I do generally concur that DHCP is not really intended to do anything other than configure the networking software in a host computer. Whatever the protocol options may have grown to include, the process for getting there is well understood and very public, so unless there is a compelling need such as inability to interoperate or insufficient options to communicate required information, we pretty much have to live with what we've got.

I'm all for (1) configuring the networking software, (2) identifying servers which can provide extended system configuration, and (3) identifying services location servers which can locate applications or generally useful services for a host computer. I think (2) and (3) are consistent with my understanding of the purposes of DHCP. I'm not so keen on providing application-specific configuration data or locating individual application-specific configuration servers, but only because I can imagine these growing in number almost without limit. A lot of this discussion really should be moved to the dhcpv4 mailing list, as most implementers follow that list.

I generally support the more general configuration of client software beyond network configuration, although I would add the restriction that some other source of this configuration *referral* data other than the DHCP server be used. What I mean by that is that instead of the DHCP server having an option for each of the dozens or thousands of applications which might like to receive configuration data from a server, that it might be a better choice to devise something akin to the Service Location protocol to provide the address of individual configuration servers -- all the DHCP server would do is to identify the "application configuration locator" server.

The proper forum for this is the DHC Working Group of IETF -- possibly a BOF session at the next working groups meeting to determine if there is interest, then either the formation of a new working group or incorporation of this work item into an existing group's charter.

[Dave Gotwisner]

Nathan is dead on with [his contention that DHCP should be able to be used to configure as much as possible in the client,] although it isn't just the implementers who desire plug and play. Many of our customers are requiring that DHCP be used in many different methods to configure our devices so the user/ administrator does not have to configure them once deployed. They want to turn on the power and let DHCP provided all useful information to configure the device. Unfortunately, with the UDP packet length, this can't really happen on complex devices, but an appropriate sub-set can be used. Other methods are better for plug and play configuration in some ways and worse in some ways (SNMP comes to mind). With plug-and-play of complex devices being configured through option 43, you run into two fundamental problems. First, the UDP maximum record size (option 57 may allow an increase in size, but not significantly). Second, option 43 is limited to 255 bytes in length, and if you encapsulate several strings (such as network pathnames) you will exceed this length for option 43.

My only other objection to option 43 (and option 18, for that matter) is that it is a bear to create an opaque option containing a heterogeneous set of option types. It makes perfect sense to send it as encapsulated data, but the tools should be smarter in how to deal with it, maybe as (expanding on Nathan's pseudo-code below):

```
     if [ vendor-class-identifier = "SUN-JAVASTATION-8MB" ]

then send option-43 {

send vendor-option-1

{ IP = 10.2.1.2,10.2.1.3 },

send vendor-option-2 { STR = "string-data"},

send vendor-option-3 { STR = "more-string-data" },

send vendor-option-30 { BOOL = True };

};

     else

        ....

endif
```

This may be harder to parse, but it would make it a lot easier to configure by a user.

I understand that option 43 should be [used for responses] as illustrated above. What I don't understand is why you can't also send different other options in this case also

send option-48 (X font server)

send option-49 (XDMCP addresses)

or other options as well, since you might want a different set of fonts loaded (for example) based upon which manufacturer's X terminal you are booting. Although the RFC says that an 43 should be given based upon option 60, it does not say that other options can't be given as well.

There are some options which a client can suggest (requested IP, requested lease time, max record size, host name (only because the server can also provide one), and option overload). I don't know why a device which has no knowledge of it's global environment can request DNS, routers, etc.

[Nathan Lane]

I should have put that (sending options other than 43) in my example since Kevin specifically mentioned a different bootfile.

Do you think we should take this over to DHC and get something going on it? We all have access to the same RFCs...I just wonder why people have such a broad interpretation of how they should be used.

[C. J. Consodine]

Any "intelligent" complication should be in the code TFTP'd to the client, not in that executed by the DHCP server. Option 60 should contain a UPC or other SKU or SKU class identifier. The logic then is to add on additional options found via option 60, subordinate to those options that would have been sent without it. One needs but a single pass through the rule base. The alternative is to add in CGI, Java or DLL like madness.

5. Acknowledgements

This document is the result of work undertaken the by DHCP working group. The authors would like to particularly acknowledge the development team from Carnegie-Mellon University whose work creating a private MIB for their DHCP server inspired the development of this proposal. In particular, many thanks to Ryan Troll who provided a great deal of useful feedback during the development of this MIB.

6. Security Considerations

Security considerations are not applicable, as this memo does not specify the interoperation of network equipment or systems, merely seeking to codify some elements of behavior not well specified by the underlying protocol.

7. References

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[RFC2132] Alexander, S. and Droms, R., "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

8. Editors' Addresses

Richard Barr Hibbs
Pacific Bell
666 Folsom Street, Room 1225
San Francisco, CA 94107-1384
USA

Phone: +1 415-545-1576
Fax: +1 415-543-3539
Email: rbhibbs@pacbell.com

Nathan Lane
Wal-Mart Stores, Inc.
702 SW 8th Street
Bentonville, AR 72716-8025
USA

Phone: +1 501-277-5786
Fax: +1 501-273-6879
Email: nathan@terminus.com

9. Full Copyright Statement

Network Working Group R. B. Hibbs, Pacific*Bell
Internet-Draft N. Lane, Wal-Mart Stores
Category: Informational Oct 1999

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.