NAME

dhcp-options - Dynamic Host Configuration Protocol options

DESCRIPTION

The Dynamic Host Configuration protocol allows the client to receive **options** from the DHCP server describing the network configuration and various services that are available on the network. When configuring **dhcpd(8)** or dhclient(8) , options must often be declared. The syntax for declaring options, and the names and formats of the options that can be declared, are documented here.

REFERENCE: OPTION STATEMENTS

DHCP *option* statements always start with the *option* keyword, followed by an option name, followed by option data. The option names and data formats are described below. It is not necessary to exhaustively specify all DHCP options - only those options which are needed by clients must be specified.

Option data comes in a variety of formats, as defined below:

The **ip-address** data type can be entered either as an explicit IP address (e.g., 239.254.197.10) or as a domain name (e.g., haagen.isc.org). When entering a domain name, be sure that that domain name resolves to a single IP address.

The **ip6-address** data specifies an IPv6 address, like ::1 or 3ffe:bbbb:aaaa:aaaa::1.

The **int32** data type specifies a signed 32-bit integer. The **uint32** data type specifies an unsigned 32-bit integer. The **int16** and **uint16** data types specify signed and unsigned 16-bit integers. The **int8** and **uint8** data types specify signed and unsigned 8-bit integers. Unsigned 8-bit integers are also sometimes referred to as octets.

The **text** data type specifies an NVT ASCII string, which must be enclosed in double quotes - for example, to specify a root-path option, the syntax would be

option root-path 10.0.1.4:/var/tmp/rootfs;

The **domain-name** data type specifies a domain name, which must not be enclosed in double quotes. The domain name is stored just as if it were a text option.

The **domain-list** data type specifies a list of domain names, enclosed in double quotes and separated by commas (example.com, foo.example.com).

The **flag** data type specifies a boolean value. Booleans can be either true or false (or on or off, if that makes more sense to you).

The **string** data type specifies either an NVT ASCII string enclosed in double quotes, or a series of octets specified in hexadecimal, separated by colons. For example:

option dhcp-client-identifier CLIENT-FOO;
or
option dhcp-client-identifier 43:4c:49:45:54:2d:46:4f:4f;

SETTING OPTION VALUES USING EXPRESSIONS

Sometimes it's helpful to be able to set the value of a DHCP option based on some value that the client has sent. To do this, you can use expression evaluation. The dhcp-eval(5) manual page describes how to write expressions. To assign the result of an evaluation to an option, define the option as follows:

option *my-option* **=** *expression* **;**

For example:

option hostname = binary-to-ascii (16, 8, -,
substring (hardware, 1, 6));

INCLUDING OPTION DEFINITIONS

Starting with 4.3.0 when ISC adds new option definitions those definitions will be included in the code based on the definition of an argument for the RFC that defines the option in includes/site.h. This provides you with a method for over-riding the ISC definitions if necessary - for example if you have previously defined the option with a different format using the mechanism from DEFINING NEW OPTIONS below.

By default all of the options are enabled. In order to disable an option you would edit the includes/site.h file and comment out the definition for the proper RFC.

STANDARD DHCPV4 OPTIONS

The documentation for the various options mentioned below is taken from the latest IETF draft document on DHCP options. Options not listed below may not yet be implemented, but it is possible to use such options by defining them in the configuration file. Please see the DEFINING NEW OPTIONS heading later in this document for more information.

Some of the options documented here are automatically generated by the DHCP server or by clients, and cannot be configured by the user. The value of such an option can be used in the configuration file of the receiving DHCP protocol agent (server or client), for example in conditional expressions. However, the value of the option cannot be used in the configuration file of the sending agent, because the value is determined only *after* the configuration file has been processed. In the following documentation, such options will be shown as not user configurable

The standard options are:

option all-subnets-local*flag***;**

This option specifies whether or not the client may assume that all subnets of the IP network to which the client is connected use the same MTU as the subnet of that network to which the client is directly connected. A value of true indicates that all subnets share the same MTU. A value of false means that the client should assume that some subnets of the directly connected network may have smaller MTUs.

option arp-cache-timeout *uint32***;**

This option specifies the timeout in seconds for ARP cache entries.

option associated-ip*ip-addr ess* [**,** *ip-address...*]**;**

This option is part of lease query. It is used to return all of the IP addresses associated with a given DHCP client.

This option is not user configurable.

option bcms-controller-address *ip-address* [**,** *ip-address...*]**;**

This option configures a list of IPv4 addresses for use as Broadcast and Multicast Controller Servers (BCMS).

option bcms-controller-names *domain-list***;**

This option contains the domain names of local Broadcast and Multicast Controller Servers (BCMS) controllers which the client may use.

option bootfile-name*text***;**

This option is used to identify a bootstrap file. If supported by the client, it should have the same effect as the **filename** declaration. BOOTP clients are unlikely to support this option. Some DHCP clients will support it, and others actually require it.

option boot-size*uint16***;**

This option specifies the length in 512-octet blocks of the default boot image for the client.

option broadcast-address*ip-addr ess***;**

This option specifies the broadcast address in use on the client's subnet. Legal values for

broadcast addresses are specified in section 3.2.1.3 of STD 3 (RFC1122).

option capwap-ac-v4 *ip-address* [**,** *ip-address ...*] **;**

A list of IPv4 addresses of CAPWAP ACs that the WTP may use. The addresses are listed in preference order.

This option is included based on RFC 5417.

option client-last-transaction-time *uint32***;**

This option is part of lease query. It allows the receiver to determine the time of the most recent access by the client. The value is a duration in seconds from when the client last communicated with the DHCP server.

This option is not user configurable.

option cookie-servers *ip-address* [**,** *ip-address...*]**;**

The cookie server option specifies a list of RFC 865 cookie servers available to the client. Servers should be listed in order of preference.

option default-ip-ttl*uint8;*

This option specifies the default time-to-live that the client should use on outgoing datagrams.

option default-tcp-ttl*uint8***;**

This option specifies the default TTL that the client should use when sending TCP segments. The minimum value is 1.

option default-url*string***;**

The format and meaning of this option is not described in any standards document, but is claimed to be in use by Apple Computer. It is not known what clients may reasonably do if supplied with this option. Use at your own risk.

option dhcp-client-identifier *string***;**

This option can be used to specify a DHCP client identifier in a host declaration, so that dhcpd can find the host record by matching against the client identifier.

Please be aware that some DHCP clients, when configured with client identifiers that are ASCII text, will prepend a zero to the ASCII text. So you may need to write:

option dhcp-client-identifier 0foo;

rather than:

option dhcp-client-identifier foo;

option dhcp-lease-time*uint32***;**

This option is used in a client request (DHCPDISCOVER or DHCPREQUEST) to allow the client to request a lease time for the IP address. In a server reply (DHCPOFFER), a DHCP server uses this option to specify the lease time it is willing to offer.

This option is not directly user configurable in the server; refer to the *max-lease-time* and *default-lease-time* server options in **dhcpd.conf(5).**

option dhcp-max-message-size*uint16***;**

This option, when sent by the client, specifies the maximum size of any response that the server sends to the client. When specified on the server, if the client did not send a dhcp-max-message-size option, the size specified on the server is used. This works for BOOTP as well as DHCP responses.

option dhcp-message*text***;**

This option is used by a DHCP server to provide an error message to a DHCP client in a DHCPNAK message in the event of a failure. A client may use this option in a DHCPDECLINE message to indicate why the client declined the offered parameters.

This option is not user configurable.

option dhcp-message-type *uint8*;

This option, sent by both client and server, specifies the type of DHCP message contained in the DHCP packet. Possible values (taken directly from RFC2132) are:

1 DHCPDISCOVER
2 DHCPOFFER
3 DHCPREQUEST
4 DHCPDECLINE
5 DHCPACK
6 DHCPNAK
7 DHCPRELEASE
8 DHCPINFORM

This option is not user configurable.

option dhcp-option-overload *uint8*;

This option is used to indicate that the DHCP ´sname´ or ´file´ fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options.

If this option is present, the client interprets the specified additional fields after it concludes interpretation of the standard option fields.

Legal values for this option are:

1 the ´file´ field is used to hold options
2 the ´sname´ field is used to hold options
3 both fields are used to hold options

This option is not user configurable.

option dhcp-parameter-request-list*uint8* [**,** *uint8...*];

This option, when sent by the client, specifies which options the client wishes the server to return. Normally, in the ISC DHCP client, this is done using the *request* statement. If this option is not specified by the client, the DHCP server will normally return every option that is valid in scope and that fits into the reply. When this option is specified on the server, the server returns the specified options. This can be used to force a client to take options that it hasn't requested, and it can also be used to tailor the response of the DHCP server for clients that may need a more limited set of options than those the server would normally return.

option dhcp-rebinding-time*uint32*;

This option specifies the number of seconds from the time a client gets an address until the client transitions to the REBINDING state.

This option is user configurable, but it will be ignored if the value is greater than or equal to the lease time.

To make DHCPv4+DHCPv6 migration easier in the future, any value configured in this option is also used as a DHCPv6 T1 (renew) time.

option dhcp-renewal-time *uint32*;

This option specifies the number of seconds from the time a client gets an address until the client transitions to the RENEWING state.

This option is user configurable, but it will be ignored if the value is greater than or equal to

the rebinding time, or lease time.

To make DHCPv4+DHCPv6 migration easier in the future, any value configured in this option is also used as a DHCPv6 T2 (rebind) time.

option dhcp-requested-address*ip-addr ess***;**

This option is used by the client in a DHCPDISCOVER to request that a particular IP address be assigned.

This option is not user configurable.

option dhcp-server-identifier *ip-address***;**

This option is used in DHCPOFFER and DHCPREQUEST messages, and may optionally be included in the DHCPACK and DHCPNAK messages. DHCP servers include this option in the DHCPOFFER in order to allow the client to distinguish between lease offers. DHCP clients use the contents of the ´server identifier´ field as the destination address for any DHCP messages unicast to the DHCP server. DHCP clients also indicate which of several lease offers is being accepted by including this option in a DHCPREQUEST message.

The value of this option is the IP address of the server.

This option is not directly user configurable. See the *server-identifier* server option in *dhcpd.conf(5).*

option domain-name*text***;**

This option specifies the domain name that client should use when resolving hostnames via the Domain Name System.

option domain-name-servers *ip-address* [**,** *ip-address...*]**;**

The domain-name-servers option specifies a list of Domain Name System (STD 13, RFC 1035) name servers available to the client. Servers should be listed in order of preference.

option domain-search *domain-list***;**

The domain-search option specifies a ´search list´ of Domain Names to be used by the client to locate not-fully-qualified domain names. The difference between this option and historic use of the domain-name option for the same ends is that this option is encoded in RFC1035 compressed labels on the wire. For example:

option domain-search example.com, sales.example.com,
eng.example.com;

option extensions-path*text***;**

This option specifies the name of a file containing additional options to be interpreted according to the DHCP option format as specified in RFC2132.

option finger-server *ip-address* [**,** *ip-address...*]**;**

The Finger server option specifies a list of Finger servers available to the client. Servers should be listed in order of preference.

option font-servers *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of X Window System Font servers available to the client. Servers should be listed in order of preference.

option geoconf-civic*string***;**

A string to hold the geoconf civic structure.

This option is included based on RFC 4776.

option host-name*string***;**

This option specifies the name of the client. The name may or may not be qualified with the local domain name (it is preferable to use the domain-name option to specify the domain name). See RFC 1035 for character set restrictions. This option is only honored by dhclient-script(8) if the hostname for the client machine is not set.

option ieee802-3-encapsulation*flag***;**

This option specifies whether or not the client should use Ethernet Version 2 (RFC 894) or IEEE 802.3 (RFC 1042) encapsulation if the interface is an Ethernet. A value of false indicates that the client should use RFC 894 encapsulation. A value of true means that the client should use RFC 1042 encapsulation.

option ien116-name-servers *ip-address* [**,** *ip-address...*];

The ien116-name-servers option specifies a list of IEN 116 name servers available to the client. Servers should be listed in order of preference.

option impress-servers *ip-address* [**,** *ip-address...*]**;**

The impress-server option specifies a list of Imagen Impress servers available to the client. Servers should be listed in order of preference.

option interface-mtu *uint16***;**

This option specifies the MTU to use on this interface. The minimum legal value for the MTU is 68.

option ip-forwarding *flag***;**

This option specifies whether the client should configure its IP layer for packet forwarding. A value of false means disable IP forwarding, and a value of true means enable IP forwarding.

option irc-server *ip-address* [**,** *ip-address...*]**;**

The IRC server option specifies a list of IRC servers available to the client. Servers should be listed in order of preference.

option log-servers *ip-address* [**,** *ip-address...*]**;**

The log-server option specifies a list of MIT-LCS UDP log servers available to the client. Servers should be listed in order of preference.

option lpr-servers *ip-address* [**,** *ip-address...*]**;**

The LPR server option specifies a list of RFC 1179 line printer servers available to the client. Servers should be listed in order of preference.

option mask-supplier*flag***;**

This option specifies whether or not the client should respond to subnet mask requests using ICMP. A value of false indicates that the client should not respond. A value of true means that the client should respond.

option max-dgram-reassembly *uint16***;**

This option specifies the maximum size datagram that the client should be prepared to reassemble. The minimum legal value is 576.

option merit-dump*text***;**

This option specifies the path-name of a file to which the client's core image should be dumped in the event the client crashes. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

option mobile-ip-home-agent *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of IP addresses indicating mobile IP home agents available to the client. Agents should be listed in order of preference, although normally there will be only one such agent.

option name-service-search *uint16* [**,** *uint6...*]**;**

This option specifies a list of name services in the order the client should attempt to use them.

This option is included based on RFC 2937.

option nds-context *string***;**

The nds-context option specifies the name of the initial Netware Directory Service for an NDS client.

option nds-servers *ip-address* [**,** *ip-address...*]**;**

The nds-servers option specifies a list of IP addresses of NDS servers.

option nds-tree-name*string***;**

The nds-tree-name option specifies NDS tree name that the NDS client should use.

option netbios-dd-server *ip-address* [**,** *ip-address...*]**;**

The NetBIOS datagram distribution server (NBDD) option specifies a list of RFC 1001/1002 NBDD servers listed in order of preference.

option netbios-name-servers *ip-address* [**,** *ip-address...*]**;**

The NetBIOS name server (NBNS) option specifies a list of RFC 1001/1002 NBNS name servers listed in order of preference. NetBIOS Name Service is currently more commonly referred to as WINS. WINS servers can be specified using the netbios-name-servers option.

option netbios-node-type *uint8***;**

The NetBIOS node type option allows NetBIOS over TCP/IP clients which are configurable to be configured as described in RFC 1001/1002. The value is specified as a single octet which identifies the client type.

Possible node types are:

1 B-node: Broadcast - no WINS

2 P-node: Peer - WINS only

4 M-node: Mixed - broadcast, then WINS

8 H-node: Hybrid - WINS, then broadcast

option netbios-scope*string***;**

The NetBIOS scope option specifies the NetBIOS over TCP/IP scope parameter for the client as specified in RFC 1001/1002. See RFC1001, RFC1002, and RFC1035 for character-set restrictions.

option netinfo-server-address *ip-address* [**,** *ip-address...*]**;**

The **netinfo-server-address** option has not been described in any RFC, but has been allocated (and is claimed to be in use) by Apple Computers. It's hard to say if the above is the correct format, or what clients might be expected to do if values were configured. Use at your own risk.

option netinfo-server-tag *text***;**

The **netinfo-server-tag** option has not been described in any RFC, but has been allocated (and is claimed to be in use) by Apple Computers. It's hard to say if the above is the correct format, or what clients might be expected to do if values were configured. Use at your own risk.

option nis-domain*text***;**

This option specifies the name of the client's NIS (Sun Network Information Services) domain. The domain is formatted as a character string consisting of characters from the NVT

ASCII character set.

option nis-servers *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of IP addresses indicating NIS servers available to the client. Servers should be listed in order of preference.

option nisplus-domain*text***;**

This option specifies the name of the client's NIS+ domain. The domain is formatted as a character string consisting of characters from the NVT ASCII character set.

option nisplus-servers *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of IP addresses indicating NIS+ servers available to the client. Servers should be listed in order of preference.

option nntp-server *ip-address* [**,** *ip-address...*]**;**

The NNTP server option specifies a list of NNTP servesr available to the client. Servers should be listed in order of preference.

option non-local-source-routing*flag***;**

This option specifies whether the client should configure its IP layer to allow forwarding of datagrams with non-local source routes (see Section 3.3.5 of [4] for a discussion of this topic). A value of false means disallow forwarding of such datagrams, and a value of true means allow forwarding.

option ntp-servers *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of IP addresses indicating NTP (RFC 5905) servers available to the client. Servers should be listed in order of preference.

option nwip-domain *string***;**

The name of the NetWare/IP domain that a NetWare/IP client should use.

option nwip-suboptions *string***;**

A sequence of suboptions for NetWare/IP clients - see RFC2242 for details. Normally this option is set by specifying specific NetWare/IP suboptions - see the NETWARE/IP SUBOPTIONS section for more information.

option option-6rd*uint8 uint8 ip6-addr ess ip-address* [**,** *ip-address ...*]**;**

This option contains infomration about the rapid deployment option. It is 8 bits of ipv4 mask length, 8 bits of 6rd prefix length, an ipv6 prefix as an ipv6 address and a list of one or more ipv4 addresses.

This option is included based on RFC 5969.

option pana-agent *ip-address* [**,** *ip-address ...*] **;**

A set of IPv4 addresses of a PAA for the client to use. The addresses are listed in preferred order.

This option is included based on RFC 5192.

option path-mtu-aging-timeout *uint32***;**

This option specifies the timeout (in seconds) to use when aging Path MTU values discovered by the mechanism defined in RFC 1191.

option path-mtu-plateau-table *uint16* [**,** *uint16...*]**;**

This option specifies a table of MTU sizes to use when performing Path MTU Discovery as defined in RFC 1191. The table is formatted as a list of 16-bit unsigned integers, ordered from smallest to largest. The minimum MTU value cannot be smaller than 68.

option pcode*text***;**

This option specifies a string suitable for the TZ variable.

This option is included based on RFC 4833.

option perform-mask-discovery *flag*;

This option specifies whether or not the client should perform subnet mask discovery using ICMP. A value of false indicates that the client should not perform mask discovery. A value of true means that the client should perform mask discovery.

option policy-filter*ip-addr ess ip-address*
[, *ip-address ip-address...*];

This option specifies policy filters for non-local source routing. The filters consist of a list of IP addresses and masks which specify destination/mask pairs with which to filter incoming source routes.

Any source routed datagram whose next-hop address does not match one of the filters should be discarded by the client.

See STD 3 (RFC1122) for further information.

option pop-server *ip-address* [, *ip-address...*];

The POP3 server option specifies a list of POP3 servers available to the client. Servers should be listed in order of preference.

option rdnss-selection*uint8 ip-addr ess ip-address domain-name*;

The rdnss-selection option specifies an 8 bit flags field, a primary and secondary ip address for the name server and a domainlist of domains for whcih the RDNSS has sepcial knowledge.

This option is included based on RFC 6731.

option resource-location-servers *ip-address* [, *ip-address...*];

This option specifies a list of RFC 887 Resource Location servers available to the client. Servers should be listed in order of preference.

option root-path*text*;

This option specifies the path-name that contains the client's root disk. The path is formatted as a character string consisting of characters from the NVT ASCII character set.

option router-discovery *flag*;

This option specifies whether or not the client should solicit routers using the Router Discovery mechanism defined in RFC 1256. A value of false indicates that the client should not perform router discovery. A value of true means that the client should perform router discovery.

option router-solicitation-address*ip-addr ess*;

This option specifies the address to which the client should transmit router solicitation requests.

option routers *ip-address* [, *ip-address...*];

The routers option specifies a list of IP addresses for routers on the client's subnet. Routers should be listed in order of preference.

option slp-directory-agent *boolean ip-address* [, *ip-address...*];

This option specifies two things: the IP addresses of one or more Service Location Protocol Directory Agents, and whether the use of these addresses is mandatory. If the initial boolean value is true, the SLP agent should just use the IP addresses given. If the value is false, the SLP agent may additionally do active or passive multicast discovery of SLP agents (see RFC2165 for details).

Please note that in this option and the slp-service-scope option, the term SLP Agent is being

used to refer to a Service Location Protocol agent running on a machine that is being configured using the DHCP protocol.

Also, please be aware that some companies may refer to SLP as NDS. If you have an NDS directory agent whose address you need to configure, the slp-directory-agent option should work.

option slp-service-scope *boolean text*;

The Service Location Protocol Service Scope Option specifies two things: a list of service scopes for SLP, and whether the use of this list is mandatory. If the initial boolean value is true, the SLP agent should only use the list of scopes provided in this option; otherwise, it may use its own static configuration in preference to the list provided in this option.

The text string should be a comma-separated list of scopes that the SLP agent should use. It may be omitted, in which case the SLP Agent will use the aggregated list of scopes of all directory agents known to the SLP agent.

option smtp-server *ip-address* [, *ip-address...*];

The SMTP server option specifies a list of SMTP servers available to the client. Servers should be listed in order of preference.

option static-routes*ip-addr ess ip-address*
[, *ip-address ip-address...*];

This option specifies a list of static routes that the client should install in its routing cache. If multiple routes to the same destination are specified, they are listed in descending order of priority.

The routes consist of a list of IP address pairs. The first address is the destination address, and the second address is the router for the destination.

The default route (0.0.0.0) is an illegal destination for a static route. To specify the default route, use the **routers** option. Also, please note that this option is not intended for classless IP routing - it does not include a subnet mask. Since classless IP routing is now the most widely deployed routing standard, this option is virtually useless, and is not implemented by any of the popular DHCP clients, for example the Microsoft DHCP client.

option streettalk-directory-assistance-server *ip-address*
[, *ip-address...*];

The StreetTalk Directory Assistance (STDA) server option specifies a list of STDA servers available to the client. Servers should be listed in order of preference.

option streettalk-server *ip-address* [, *ip-address...*];

The StreetTalk server option specifies a list of StreetTalk servers available to the client. Servers should be listed in order of preference.

option subnet-mask *ip-address*;

The subnet mask option specifies the client's subnet mask as per RFC 950. If no subnet mask option is provided anywhere in scope, as a last resort dhcpd will use the subnet mask from the subnet declaration for the network on which an address is being assigned. However, *any* subnet-mask option declaration that is in scope for the address being assigned will override the subnet mask specified in the subnet declaration.

option subnet-selection*ip-addr ess*;

Sent by the client if an address is required in a subnet other than the one that would normally be selected (based on the relaying address of the connected subnet the request is obtained from). See RFC3011. Note that the option number used by this server is 118; this has not always been the defined number, and some clients may use a different value. Use of this option should be regarded as slightly experimental!

This option is not user configurable in the server.

option swap-server *ip-address*;

This specifies the IP address of the client's swap server.

option tcp-keepalive-garbage *flag*;

This option specifies whether or not the client should send TCP keepalive messages with an octet of garbage for compatibility with older implementations. A value of false indicates that a garbage octet should not be sent. A value of true indicates that a garbage octet should be sent.

option tcp-keepalive-interval *uint32*;

This option specifies the interval (in seconds) that the client TCP should wait before sending a keepalive message on a TCP connection. The time is specified as a 32-bit unsigned integer. A value of zero indicates that the client should not generate keepalive messages on connections unless specifically requested by an application.

option tcode*text*;

This option specifies a name of a zone entry in the TZ database.

This option is included based on RFC 4833.

option tftp-server-name *text*;

This option is used to identify a TFTP server and, if supported by the client, should have the same effect as the **server-name** declaration. BOOTP clients are unlikely to support this option. Some DHCP clients will support it, and others actually require it.

option time-offset *int32*;

The time-offset option specifies the offset of the client's subnet in seconds from Coordinated Universal Time (UTC).

option time-servers *ip-address* [, *ip-address*...];

The time-server option specifies a list of RFC 868 time servers available to the client. Servers should be listed in order of preference.

option trailer-encapsulation*flag*;

This option specifies whether or not the client should negotiate the use of trailers (RFC 893 [14]) when using the ARP protocol. A value of false indicates that the client should not attempt to use trailers. A value of true means that the client should attempt to use trailers.

option uap-servers *text*;

This option specifies a list of URLs, each pointing to a user authentication service that is capable of processing authentication requests encapsulated in the User Authentication Protocol (UAP). UAP servers can accept either HTTP 1.1 or SSLv3 connections. If the list includes a URL that does not contain a port component, the normal default port is assumed (i.e., port 80 for http and port 443 for https). If the list includes a URL that does not contain a path component, the path /uap is assumed. If more than one URL is specified in this list, the URLs are separated by spaces.

option user-class*string*;

This option is used by some DHCP clients as a way for users to specify identifying information to the client. This can be used in a similar way to the vendor-class-identifier option, but the value of the option is specified by the user, not the vendor. Most recent DHCP clients have a way in the user interface to specify the value for this identifier, usually as a text string.

option v4-access-domain*domain-name*;

The domain name associated with the access network for use with LIS Discovery.

This option is included based on RFC 5986.

option v4-lost*domain-name***;**

The domain name of the LoST server for the client to use.

This option is included based on RFC 5223.

option vendor-class-identifier *string***;**

This option is used by some DHCP clients to identify the vendor type and possibly the configuration of a DHCP client. The information is a string of bytes whose contents are specific to the vendor and are not specified in a standard. To see what vendor class identifier clients are sending, you can write the following in your DHCP server configuration file:

set vendor-string = option vendor-class-identifier;

This will result in all entries in the DHCP server lease database file for clients that sent vendor-class-identifier options having a set statement that looks something like this:

set vendor-string = SUNW.Ultra-5_10;

The vendor-class-identifier option is normally used by the DHCP server to determine the options that are returned in the **vendor-encapsulated-options** option. Please see the VENDOR ENCAPSULATED OPTIONS section later in this manual page for further information.

option vendor-encapsulated-options *string***;**

The **vendor-encapsulated-options** option can contain either a single vendor-specific value or one or more vendor-specific suboptions. This option is not normally specified in the DHCP server configuration file - instead, a vendor class is defined for each vendor, vendor class suboptions are defined, values for those suboptions are defined, and the DHCP server makes up a response on that basis.

Some default behaviours for well-known DHCP client vendors (currently, the Microsoft Windows 2000 DHCP client) are configured automatically, but otherwise this must be configured manually - see the VENDOR ENCAPSULATED OPTIONS section later in this manual page for details.

option vivso*string***;**

The **vivso** option can contain multiple separate options, one for each 32-bit Enterprise ID. Each Enterprise-ID discriminated option then contains additional options whose format is defined by the vendor who holds that ID. This option is usually not configured manually, but rather is configured via intervening option definitions. Please also see the VENDOR ENCAPSULATED OPTIONS section later in this manual page for details.

option www-server *ip-address* [**,** *ip-address...*]**;**

The WWW server option specifies a list of WWW servers available to the client. Servers should be listed in order of preference.

option x-display-manager *ip-address* [**,** *ip-address...*]**;**

This option specifies a list of systems that are running the X Window System Display Manager and are available to the client. Addresses should be listed in order of preference.

RELAY AGENT INFORMATION OPTION

An IETF draft, draft-ietf-dhc-agent-options-11.txt, defines a series of encapsulated options that a relay agent can add to a DHCP packet when relaying it to the DHCP server. The server can then make address allocation decisions (or whatever other decisions it wants) based on these options. The server also returns these options in any replies it sends through the relay agent, so that the relay agent can use the information in these options for delivery or accounting purposes.

The current draft defines two options. To reference these options in the dhcp server, specify the option space name, agent, followed by a period, followed by the option name. It is not normally

useful to define values for these options in the server, although it is permissible. These options are not supported in the client.

option agent.circuit-id *string*;

The circuit-id suboption encodes an agent-local identifier of the circuit from which a DHCP client-to-server packet was received. It is intended for use by agents in relaying DHCP responses back to the proper circuit. The format of this option is currently defined to be vendor-dependent, and will probably remain that way, although the current draft allows for the possibility of standardizing the format in the future.

option agent.remote-id *string*;

The remote-id suboption encodes information about the remote host end of a circuit. Examples of what it might contain include caller ID information, username information, remote ATM address, cable modem ID, and similar things. In principal, the meaning is not well-specified, and it should generally be assumed to be an opaque object that is administratively guaranteed to be unique to a particular remote end of a circuit.

option agent.DOCSIS-device-class *uint32*;

The DOCSIS-device-class suboption is intended to convey information about the host endpoint, hardware, and software, that either the host operating system or the DHCP server may not otherwise be aware of (but the relay is able to distinguish). This is implemented as a 32-bit field (4 octets), each bit representing a flag describing the host in one of these ways. So far, only bit zero (being the least significant bit) is defined in RFC3256. If this bit is set to one, the host is considered a CPE Controlled Cable Modem (CCCM). All other bits are reserved.

option agent.link-selection *ip-address*;

The link-selection suboption is provided by relay agents to inform servers what subnet the client is actually attached to. This is useful in those cases where the giaddr (where responses must be sent to the relay agent) is not on the same subnet as the client. When this option is present in a packet from a relay agent, the DHCP server will use its contents to find a subnet declared in configuration, and from here take one step further backwards to any shared-network the subnet may be defined within; the client may be given any address within that shared network, as normally appropriate.

THE CLIENT FQDN SUBOPTIONS

The Client FQDN option, currently defined in the Internet Draft draft-ietf-dhc-fqdn-option-00.txt is not a standard yet, but is in sufficiently wide use already that we have implemented it. Due to the complexity of the option format, we have implemented it as a suboption space rather than a single option. In general this option should not be configured by the user - instead it should be used as part of an automatic DNS update system.

option fqdn.no-client-update *flag*;

When the client sends this, if it is true, it means the client will not attempt to update its A record. When sent by the server to the client, it means that the client *should not* update its own A record.

option fqdn.server-update *flag*;

When the client sends this to the server, it is requesting that the server update its A record. When sent by the server, it means that the server has updated (or is about to update) the client's A record.

option fqdn.encoded *flag*;

If true, this indicates that the domain name included in the option is encoded in DNS wire format, rather than as plain ASCII text. The client normally sets this to false if it doesn't support DNS wire format in the FQDN option. The server should always send back the same

value that the client sent. When this value is set on the configuration side, it controls the format in which the *fqdn.fqdn* suboption is encoded.

option fqdn.rcode1 *flag*;

option fqdn.rcode2 *flag*;

These options specify the result of the updates of the A and PTR records, respectively, and are only sent by the DHCP server to the DHCP client. The values of these fields are those defined in the DNS protocol specification.

option fqdn.fqdn *text*;

Specifies the domain name that the client wishes to use. This can be a fully-qualified domain name, or a single label. If there is no trailing ´.´ character in the name, it is not fully-qualified, and the server will generally update that name in some locally-defined domain.

option fqdn.hostname *--never set--*;

This option should never be set, but it can be read back using the **option** and **config-option** operators in an expression, in which case it returns the first label in the **fqdn.fqdn** suboption - for example, if the value of **fqdn.fqdn** is foo.example.com., then **fqdn.hostname** will be foo.

option fqdn.domainname *--never set--*;

This option should never be set, but it can be read back using the **option** and **config-option** operators in an expression, in which case it returns all labels after the first label in the **fqdn.fqdn** suboption - for example, if the value of **fqdn.fqdn** is foo.example.com., then **fqdn.domainname** will be example.com.. If this suboption value is not set, it means that an unqualified name was sent in the **fqdn** option, or that no **fqdn** option was sent at all.

If you wish to use any of these suboptions, we strongly recommend that you refer to the Client FQDN option draft (or standard, when it becomes a standard) - the documentation here is sketchy and incomplete in comparison, and is just intended for reference by people who already understand the Client FQDN option specification.

THE NETWARE/IP SUBOPTIONS

RFC2242 defines a set of encapsulated options for Novell NetWare/IP clients. To use these options in the dhcp server, specify the option space name, nwip, followed by a period, followed by the option name. The following options can be specified:

option nwip.nsq-broadcast *flag*;

If true, the client should use the NetWare Nearest Server Query to locate a NetWare/IP server. The behaviour of the Novell client if this suboption is false, or is not present, is not specified.

option nwip.preferred-dss *ip-address* [**,** *ip-address...*];

This suboption specifies a list of up to five IP addresses, each of which should be the IP address of a NetWare Domain SAP/RIP server (DSS).

option nwip.nearest-nwip-server *ip-address* [**,** *ip-address...*];

This suboption specifies a list of up to five IP addresses, each of which should be the IP address of a Nearest NetWare IP server.

option nwip.autoretries *uint8*;

Specifies the number of times that a NetWare/IP client should attempt to communicate with a given DSS server at startup.

option nwip.autoretry-secs *uint8*;

Specifies the number of seconds that a Netware/IP client should wait between retries when attempting to establish communications with a DSS server at startup.

option nwip.nwip-1-1 *uint8*;

If true, the NetWare/IP client should support NetWare/IP version 1.1 compatibility. This is only needed if the client will be contacting Netware/IP version 1.1 servers.

option nwip.primary-dss *ip-address*;

Specifies the IP address of the Primary Domain SAP/RIP Service server (DSS) for this NetWare/IP domain. The NetWare/IP administration utility uses this value as Primary DSS server when configuring a secondary DSS server.

STANDARD DHCPV6 OPTIONS

DHCPv6 options differ from DHCPv4 options partially due to using 16-bit code and length tags, but semantically zero-length options are legal in DHCPv6, and multiple options are treated differently. Whereas in DHCPv4 multiple options would be concatenated to form one option, in DHCPv6 they are expected to be individual instantiations. Understandably, many options are not allowed to have multiple instances in a packet - normally these are options which are digested by the DHCP protocol software, and not by users or applications.

option dhcp6.client-id *string*;

This option specifies the client's DUID identifier. DUIDs are similar but different from DHCPv4 client identifiers - there are documented duid types:

duid-llt

duid-en

duid-ll

This value should not be configured, but rather is provided by clients and treated as an opaque identifier key blob by servers.

option dhcp6.server-id *string*;

This option specifies the server's DUID identifier. One may use this option to configure an opaque binary blob for your server's identifier.

option dhcp6.ia-na*string*;

The Identity Association for Non-temporary Addresses (ia-na) carries assigned addresses that are not temporary addresses for use by the DHCPv6 client. This option is produced by the DHCPv6 server software, and should not be configured.

option dhcp6.ia-ta*string*;

The Identity Association for Temporary Addresses (ia-ta) carries temporary addresses, which may change upon every renewal. There is no support for this in the current DHCPv6 software.

option dhcp6.ia-addr*string*;

The Identity Association Address option is encapsulated inside ia-na or ia-ta options in order to represent addresses associated with those IA's. These options are manufactured by the software, so should not be configured.

option dhcp6.oro*uint16* [**,** *uint16*, ...];

The Option Request Option (ORO) is the DHCPv6 equivalent of the parameter-request-list. Clients supply this option to ask servers to reply with options relevant to their needs and use. This option must not be directly configured, the request syntax in dhclient.conf (5) should be used instead.

option dhcp6.preference*uint8*;

The **preference** option informs a DHCPv6 client which server is ´preferred´ for use on a given subnet. This preference is only applied during the initial stages of configuration - once a client is bound to an IA, it will remain bound to that IA until it is no longer valid or has

expired. This value may be configured on the server, and is digested by the client software.

option dhcp6.elapsed-time*uint16***;**

The **elapsed-time** option is constructed by the DHCPv6 client software, and is potentially consumed by intermediaries. This option should not be configured.

option dhcp6.relay-msg *string***;**

The **relay-msg** option is constructed by intervening DHCPv6 relay agent software. This option is entirely used by protocol software, and is not meant for user configuration.

option dhcp6.unicast*ip6-addr ess***;**

The **unicast** option is provided by DHCPv6 servers which are willing (or prefer) to receive Renew packets from their clients by exchanging UDP unicasts with them. Normally, DHCPv6 clients will multicast their Renew messages. This may be configured on the server, and should be configured as an address the server is ready to reply to.

option dhcp6.status-code*status-c ode* [*string*] **;**

The **status-code** option is provided by DHCPv6 servers to inform clients of error conditions during protocol communication. This option is manufactured and digested by protocol software, and should not be configured.

option dhcp6.rapid-commit ;

The **rapid-commit** option is a zero-length option that clients use to indicate their desire to enter into rapid-commit with the server.

option dhcp6.vendor-opts *string***;**

The **vendor-opts** option is actually an encapsulated sub-option space, in which each Vendor-specific Information Option (VSIO) is identified by a 32-bit Enterprise-ID number. The encapsulated option spaces within these options are defined by the vendors.

To make use of this option, the best way is to examine the section titled VENDOR ENCAPSULATED OPTIONS below, in particular the bits about the vsio option space.

option dhcp6.interface-id *string***;**

The **interface-id** option is manufactured by relay agents, and may be used to guide configuration differentiating clients by the interface they are remotely attached to. It does not make sense to configure a value for this option, but it may make sense to inspect its contents.

option dhcp6.reconf-msg*dhcpv6-message***;**

The **reconf-msg** option is manufactured by servers, and sent to clients in Reconfigure messages to inform them of what message the client should Reconfigure using. There is no support for DHCPv6 Reconfigure extensions, and this option is documented informationally only.

option dhcp6.reconf-accept ;

The **reconf-accept** option is included by DHCPv6 clients that support the Reconfigure extentions, advertising that they will respond if the server were to ask them to Reconfigure. There is no support for DHCPv6 Reconfigure extensions, and this option is documented informationally only.

option dhcp6.sip-servers-names *domain-list***;**

The **sip-servers-names** option allows SIP clients to locate a local SIP server that is to be used for all outbound SIP requests, a so-calledoutbound proxy server. If you wish to use manually entered IPv6 addresses instead, please see the **sip-servers-addresses** option below.

option dhcp6.sip-servers-addresses *ip6-address* [**,** *ip6-address ...*] **;**

The **sip-servers-addresses** option allows SIP clients to locate a local SIP server that is to be used for all outbound SIP requests, a so-called outbound proxy servers. If you wish to use

domain names rather than IPv6 addresses, please see the **sip-servers-names** option above.

option dhcp6.name-servers *ip6-address* [**,** *ip6-address ...*] **;**

The **name-servers** option instructs clients about locally available recursive DNS servers. It is easiest to describe this as the nameserver line in /etc/resolv.conf.

option dhcp6.domain-search *domain-list***;**

The **domain-search** option specifies the client's domain search path to be applied to recursive DNS queries. It is easiest to describe this as the search line in /etc/resolv.conf.

option dhcp6.ia-pd*string***;**

The **ia-pd** option is manufactured by clients and servers to create a Prefix Delegation binding - to delegate an IPv6 prefix to the client. It is not directly edited in dhcpd.conf(5) or dhclient.conf(5), but rather is manufactured and consumed by the software.

option dhcp6.ia-prefix*string***;**

The **ia-prefix** option is placed inside **ia-pd** options in order to identify the prefix(es) allocated to the client. It is not directly edited in dhcpd.conf(5) or dhclient.conf(5), but rather is manufactured and consumed by the software.

option dhcp6.nis-servers *ip6-address* [**,** *ip6-address ...*] **;**

The **nis-servers** option identifies, in order, NIS servers available to the client.

option dhcp6.nisp-servers *ip6-address* [**,** *ip6-address ...*] **;**

The **nisp-servers** option identifies, in order, NIS+ servers available to the client.

option nis-domain-name*domain-list***;**

The **nis-domain-name** option specifies the NIS domain name the client is expected to use, and is related to the **nis-servers** option.

option dhcp6.nis-domain-name*domain-name***;**

The **dhcp6.nis-domain-name** option specifies NIS domain name the client is expected to use, and is related to **dhcp6.nis-servers** option.

option nisp-domain-name*domain-list***;**

The **nisp-domain-name** option specifies the NIS+ domain name the client is expected to use, and is related to the **nisp-servers** option.

option dhcp6.nisp-domain-name*domain-name***;**

The **dhcp6.nis-domain-name** option specifies NIS+ domain name the client is expected to use, and is related to **dhcp6.nisp-servers** option.

option dhcp6.sntp-servers *ip6-address* [**,** *ip6-address ...*] **;**

The **sntp-servers** option specifies a list of local SNTP servers available for the client to synchronize their clocks.

option dhcp6.info-refresh-time*uint32***;**

The **info-refresh-time** option gives DHCPv6 clients using Information-request messages a hint as to how long they should between refreshing the information they were given. Note that this option will only be delivered to the client, and be likely to affect the client's behaviour, if the client requested the option.

option dhcp6.bcms-server-d *domain-list***;**

The **bcms-server-d** option contains the domain names of local BCMS (Broadcast and Multicast Control Services) controllers which the client may use.

option dhcp6.bcms-server-a *ip6-address* [**,** *ip6-address ...*] **;**

The **bcms-server-a** option contains the IPv6 addresses of local BCMS (Broadcast and Multicast Control Services) controllers which the client may use.

option dhcp6.geoconf-civic*string***;**

A string to hold the geoconf civic structure.

This option is included based on RFC 4776.

option dhcp6.remote-id*string***;**

The **remote-id** option is constructed by relay agents, to inform the server of details pertaining to what the relay knows about the client (such as what port it is attached to, and so forth). The contents of this option have some vendor-specific structure (similar to VSIO), but we have chosen to treat this option as an opaque field.

option dhcp6.subscriber-id*string***;**

The **subscriber-id** option is an opaque field provided by the relay agent, which provides additional information about the subscriber in question. The exact contents of this option depend upon the vendor and/or the operator's configuration of the remote device, and as such is an opaque field.

option dhcp6.fqdn*string***;**

The **fqdn** option is normally constructed by the client or server, and negotiates the client's Fully Qualified Domain Name, as well as which party is responsible for Dynamic DNS Updates. See the section on the Client FQDN SubOptions for full details (the DHCPv4 and DHCPv6 FQDN options use the same fqdn. encapsulated space, so are in all ways identical).

option dhcp6.pana-agent *ip6-address* [**,** *ip6-address ...*] **;**

A set of IPv6 addresses of a PAA for the client to use. The addresses are listed in preferred order.

This option is included based on RFC 5192.

option dhcp6.new-posix-timezone*text***;**

This option specifies a string suitable for the TZ variable.

This option is included based on RFC 4833.

option dhcp6.new-tzdb-timezone*text***;**

This option specifies a name of a zone entry in the TZ database.

This option is included based on RFC 4833.

option dhcp6.ero*uint16* [**,** *uint16 ...*] **;**

A list of the options requested by the relay agent.

This option is included based on RFC 4994.

option dhcp6.lq-query*string***;**

The **lq-query** option is used internally for lease query.

option dhcp6.client-data *string***;**

The **client-data** option is used internally for lease query.

option dhcp6.clt-time*uint32***;**

The **clt-time** option is used internally for lease query.

option dhcp6.lq-relay-data *ip6-address string***;**

The **lq-relay-data** option is used internally for lease query.

option dhcp6.lq-client-link *ip6-address* [**,** *ip6-address ...*] **;**

The **lq-client-link** option is used internally for lease query.

option dhcp6.v6-lost*domain-name***;**

The domain name of the LoST server for the client to use.

This option is included based on RFC 5223.

option dhcp6.capwap-ac-v6 *ip6-address* [**,** *ip6-address ...*] **;**

A list of IPv6 addresses of CAPWAP ACs that the WTP may use. The addresses are listed in preference order.

This option is included based on RFC 5417.

option dhcp6.relay-id *string***;**

The DUID for the relay agent.

This option is included based on RFC 5460.

option dhcp6.v6-access-domain*domain-name***;**

The domain name associated with the access network for use with LIS Discovery.

This option is included based on RFC5986.

option dhcp6.sip-ua-cs-list*domain-list***;**

The list of domain names in the SIP User Agent Configuration Service Domains.

This option is included based on RFC 6011.

option dhcp6.bootfile-url*text***;**

The URL for a boot file.

This option is included based on RFC 5970.

option dhcp6.bootfile-param*string***;**

A string for the parameters to the bootfile. See RFC 5970 for more description of the layout of the parameters within the string.

This option is included based on RFC 5970.

option dhcp6.client-arch-type *uint16* [**,** *uint16 ...*] **;**

A list of one or more architecture types described as 16 bit values.

This option is included based on RFC 5970.

option dhcp6.nii*uint8 uint8 uint8***;**

The client network interface identitier option supplies information about a client's level of UNDI support. The values are, in order, the type, the major value and the minor value.

This option is included based on RFC5970.

option dhcp6.aftr-name*domain-name***;**

A domain name of the AFTR tunnel endpoint.

This option is included based on RFC 6334.

option dhcp6.erp-local-domain-name*domain-name***;**

A domain name for the ERP domain.

This option is included based on RFC 6440.

option dhcp6.rdnss-selection*ip6-addr ess uint8 domain-name***;**

RDNSS information consists of an IPv6 address of RDNSS, an 8 bit flags field and a domain-list of domains for which the RDNSS has special knowledge.

This option is included based on RFC 6731.

option dhcp6.client-linklayer-addr *string*;

A client link-layer address. The first two bytes must be the type of the link-layer followed by the address itself.

This option is included based on RFC 6939.

option dhcp6.link-address*ip6-addr ess*;

An IPv6 address used by a relay agent to indicate to the server the link on which the client is located.

This option is included based on RFC 6977.

option dhcp6.solmax-rt*uint32*;

A value to override the default for SOL_MAX_RT. This is a 32 bit value.

This option is included based on RFC 7083.

option dhcp6.inf-max-rt*uint32*;

A value to override the default for INF_MAX_RT. This is a 32 bit value.

This option is included based on RFC 7083.

ACCESSING DHCPV6 RELAY OPTIONS

v6relay (relay-number option This option allows access to an option that has been added to a packet by a relay agent. Relay-number value selects the relay to examine and option is the option to find. In DHCPv6 each relay encapsulates the entire previous message into an option, adds its own options (if any) and sends the result onwards. The RFC specifies a limit of 32 hops. A relay-number of 0 is a no-op and means don't look at the relays. 1 is the relay that is closest to the client, 2 would be the next in from the client and so on. Any value greater than the max number of hops is which is closest to the server independent of number. To use this option in a class statement you would have something like this:

match if v6relay(1, option dhcp6.subscriber-id) = client_1;

DEFINING NEW OPTIONS

The Internet Systems Consortium DHCP client and server provide the capability to define new options. Each DHCP option has a name, a code, and a structure. The name is used by you to refer to the option. The code is a number, used by the DHCP server and client to refer to an option. The structure describes what the contents of an option looks like.

To define a new option, you need to choose a name for it that is not in use for some other option - for example, you can't use host-name because the DHCP protocol already defines a host-name option, which is documented earlier in this manual page. If an option name doesn't appear in this manual page, you can use it, but it's probably a good idea to put some kind of unique string at the beginning so you can be sure that future options don't take your name. For example, you might define an option, local-host-name, feeling some confidence that no official DHCP option name will ever start with local.

Once you have chosen a name, you must choose a code. All codes between 224 and 254 are reserved as ´site-local´ DHCP options, so you can pick any one of these for your site (not for your product/application). In RFC3942, site-local space was moved from starting at 128 to starting at 224. In practice, some vendors have interpreted the protocol rather loosely and have used option code values greater than 128 themselves. There's no real way to avoid this problem, and it was thought to be unlikely to cause too much trouble in practice. If you come across a vendor-documented option code in either the new or old site-local spaces, please contact your vendor and inform them about rfc3942.

The structure of an option is simply the format in which the option data appears. The ISC DHCP server currently supports a few simple types, like integers, booleans, strings and IP addresses, and

it also supports the ability to define arrays of single types or arrays of fixed sequences of types.

New options are declared as follows:

option *new-name* **code** *new-code* **=** *definition* **;**

The values of *new-name* and *new-code* should be the name you have chosen for the new option and the code you have chosen. The *definition* should be the definition of the structure of the option.

The following simple option type definitions are supported:

BOOLEAN

option *new-name* **code** *new-code* **= boolean ;**

An option of type boolean is a flag with a value of either on or off (or true or false). So an example use of the boolean type would be:

option use-zephyr code 180 = boolean;
option use-zephyr on;

INTEGER

option *new-name* **code** *new-code* **=** *sign* **integer** *width* **;**

The *sign* token should either be blank, *unsigned* or *signed*. The width can be either 8, 16 or 32, and refers to the number of bits in the integer. So for example, the following two lines show a definition of the sql-connection-max option and its use:

option sql-connection-max code 192 = unsigned integer 16;
option sql-connection-max 1536;

IP-ADDRESS

option *new-name* **code** *new-code* **= ip-address ;**

An option whose structure is an IP address can be expressed either as a domain name or as a dotted quad. So the following is an example use of the ip-address type:

option sql-server-address code 193 = ip-address;
option sql-server-address sql.example.com;

IP6-ADDRESS

option *new-name* **code** *new-code* **= ip6-address ;**

An option whose structure is an IPv6 address must be expressed as a valid IPv6 address. The following is an example use of the ip6-address type:

option dhcp6.some-server code 1234 = array of ip6-address;
option dhcp6.some-server 3ffe:bbbb:aaaa:aaaa::1, 3ffe:bbbb:aaaa:aaaa::2;

TEXT

option *new-name* **code** *new-code* **= text ;**

An option whose type is text will encode an ASCII text string. For example:

option sql-default-connection-name code 194 = text;
option sql-default-connection-name PRODZA;

DATA STRING

option *new-name* **code** *new-code* **= string ;**

An option whose type is a data string is essentially just a collection of bytes, and can be specified either as quoted text, like the text type, or as a list of hexadecimal contents separated by colons whose values must be between 0 and FF. For example:

option sql-identification-token code 195 = string;
option sql-identification-token 17:23:19:a6:42:ea:99:7c:22;

DOMAIN-LIST

option *new-name* **code** *new-code* **= domain-list** [**compressed**] **;**

An option whose type is **domain-list** is an RFC1035 formatted (on the wire, DNS Format) list of domain names, separated by root labels. The optional **compressed** keyword indicates if the option should be compressed relative to the start of the option contents (not the packet contents).

When in doubt, omit the **compressed** keyword. When the software receives an option that is compressed and the **compressed** keyword is omitted, it will still decompress the option (relative to the option contents field). The keyword only controls whether or not transmitted packets are compressed.

Note that when **domain-list** formatted options are output as environment variables to dhclient-script(8), the standard DNS -escape mechanism is used: they are decimal. This is appropriate for direct use in eg /etc/resolv.conf.

ENCAPSULATION

option *new-name* **code** *new-code* **= encapsulate** *identifier* **;**

An option whose type is **encapsulate** will encapsulate the contents of the option space specified in *identifier*. Examples of encapsulated options in the DHCP protocol as it currently exists include the vendor-encapsulated-options option, the netware-suboptions option and the relay-agent-information option.

option space local;
option local.demo code 1 = text;
option local-encapsulation code 197 = encapsulate local;
option local.demo demo;

ARRAYS

Options can contain arrays of any of the above types except for the text and data string types, which aren't currently supported in arrays. An example of an array definition is as follows:

option kerberos-servers code 200 = array of ip-address;
option kerberos-servers 10.20.10.1, 10.20.11.1;

RECORDS

Options can also contain data structures consisting of a sequence of data types, which is sometimes called a record type. For example:

option contrived-001 code 201 = { boolean, integer 32, text };
option contrived-001 on 1772 contrivance;

It's also possible to have options that are arrays of records, for example:

option new-static-routes code 201 = array of {
ip-address, ip-address, ip-address, integer 8 };
option static-routes
10.0.0.0 255.255.255.0 net-0-rtr.example.com 1,
10.0.1.0 255.255.255.0 net-1-rtr.example.com 1,
10.2.0.0 255.255.224.0 net-2-0-rtr.example.com 3;

VENDOR ENCAPSULATED OPTIONS

The DHCP protocol defines the **vendor-encapsulated-options** option, which allows vendors to define their own options that will be sent encapsulated in a standard DHCP option. It also defines the **Vendor Identified Vendor Sub Options**option (VIVSO), and the DHCPv6 protocol

defines the **Vendor-specific Information Option**(VSIO). The format of all of these options is usually internally a string of options, similarly to other normal DHCP options. The VIVSO and VSIO options differ in that they contain options that correspond to vendor Enterprise-ID numbers (assigned by IANA), which then contain options according to each Vendor's specifications. You will need to refer to your vendor's documentation in order to form options to their specification.

The value of these options can be set in one of two ways. The first way is to simply specify the data directly, using a text string or a colon-separated list of hexadecimal values. For help in forming these strings, please refer to **RFC2132** for the DHCPv4 **Vendor Specific Information Option**, **RFC3925** for the DHCPv4 **Vendor Identified Vendor Sub Options**, or **RFC3315** for the DHCPv6 **Vendor-specific Information Option**. For example:

option vendor-encapsulated-options
2:4:
AC:11:41:1:
3:12:
73:75:6e:64:68:63:70:2d:73:65:72:76:65:72:31:37:2d:31:
4:12:
2f:65:78:70:6f:72:74:2f:72:6f:6f:74:2f:69:38:36:70:63;
option vivso
00:00:09:bf:0E:
01:0c:
48:65:6c:6c:6f:20:77:6f:72:6c:64:21;
option dhcp6.vendor-opts
00:00:09:bf:
00:01:00:0c:
48:65:6c:6c:6f:20:77:6f:72:6c:64:21;

The second way of setting the value of these options is to have the DHCP server generate a vendor-specific option buffer. To do this, you must do four things: define an option space, define some options in that option space, provide values for them, and specify that that option space should be used to generate the relevant option.

To define a new option space in which vendor options can be stored, use the option space statement:

option space *name* [[**code width** *number*] [**length width** *number*] [**hash size** *number*]] **;**

Where the numbers following **code width**, **length width**, and **hash size** respectively identify the number of bytes used to describe option codes, option lengths, and the size in buckets of the hash tables to hold options in this space (most DHCPv4 option spaces use 1 byte codes and lengths, which is the default, whereas most DHCPv6 option spaces use 2 byte codes and lengths).

The code and length widths are used in DHCP protocol - you must configure these numbers to match the applicable option space you are configuring. They each default to 1. Valid values for code widths are 1, 2 or 4. Valid values for length widths are 0, 1 or 2. Most DHCPv4 option spaces use 1 byte codes and lengths, which is the default, whereas most DHCPv6 option spaces use 2 byte codes and lengths. A zero-byte length produces options similar to the DHCPv6 Vendor-specific Information Option - but not their contents!

The hash size defaults depend upon the **code width** selected, and may be 254 or 1009. Valid values range between 1 and 65535. Note that the higher you configure this value, the more memory will be used. It is considered good practice to configure a value that is slightly larger than the estimated number of options you plan to configure within the space. Previous versions of ISC DHCP (up to and including DHCP 3.0.*), this value was fixed at 9973.

The name can then be used in option definitions, as described earlier in this document. For example:

```
option space SUNW code width 1 length width 1 hash size 3;
option SUNW.server-address code 2 = ip-address;
option SUNW.server-name code 3 = text;
option SUNW.root-path code 4 = text;

option space ISC code width 1 length width 1 hash size 3;
option ISC.sample code 1 = text;
option vendor.ISC code 2495 = encapsulate vivso-sample;
option vendor-class.ISC code 2495 = text;

option ISC.sample configuration text here;
option vendor-class.ISC vendor class here;

option space docsis code width 2 length width 2 hash size 17;
option docsis.tftp-servers code 32 = array of ip6-address;
option docsis.cablelabs-configuration-file code 33 = text;
option docsis.cablelabs-syslog-servers code 34 = array of ip6-address;
option docsis.device-id code 36 = string;
option docsis.time-servers code 37 = array of ip6-address;
option docsis.time-offset code 38 = signed integer 32;
option vsio.docsis code 4491 = encapsulate docsis;
```

Once you have defined an option space and the format of some options, you can set up scopes that define values for those options, and you can say when to use them. For example, suppose you want to handle two different classes of clients. Using the option space definition shown in the previous example, you can send different option values to different clients based on the vendor-class-identifier option that the clients send, as follows:

```
class vendor-classes {
match option vendor-class-identifier;
}

subclass vendor-classes SUNW.Ultra-5_10 {
vendor-option-space SUNW;
option SUNW.root-path /export/root/sparc;
}

subclass vendor-classes SUNW.i86pc {
vendor-option-space SUNW;
option SUNW.root-path /export/root/i86pc;
}

option SUNW.server-address 172.17.65.1;
option SUNW.server-name sundhcp-server17-1;

option vivso-sample.sample Hello world!;

option docsis.tftp-servers ::1;
```

As you can see in the preceding example, regular scoping rules apply, so you can define values that are global in the global scope, and only define values that are specific to a particular class in the local scope. The **vendor-option-space** declaration tells the DHCP server to use options in the SUNW option space to construct the DHCPv4 **vendor-encapsulated-options** option. This is a limitation of that option - the DHCPv4 VIVSO and the DHCPv6 VSIO options can have multiple vendor definitions all at once (even transmitted to the same client), so it is not necessary to configure this.

SEE ALSO

dhcpd.conf(5), dhcpd.leases(5), dhclient.conf(5), dhcp-eval(5), dhcpd(8), dhclient(8), RFC2132, RFC2131, RFC3046, RFC3315.

AUTHOR

Information about Internet Systems Consortium can be found at https://www.isc.org.