NAME

dhclient.conf - DHCP client configuration file

DESCRIPTION

The dhclient.conf file contains configuration information for *dhclient,* the Internet Systems Consortium DHCP Client.

The dhclient.conf file is a free-form ASCII text file. It is parsed by the recursive-descent parser built into dhclient. The file may contain extra tabs and newlines for formatting purposes. Keywords in the file are case-insensitive. Comments may be placed anywhere within the file (except within quotes). Comments begin with the # character and end at the end of the line.

The dhclient.conf file can be used to configure the behaviour of the client in a wide variety of ways: protocol timing, information requested from the server, information required of the server, defaults to use if the server does not provide certain information, values with which to override information provided by the server, or values to prepend or append to information provided by the server. The configuration file can also be preinitialized with addresses to use on networks that don't have DHCP servers.

PROTOCOL TIMING

The timing behaviour of the client need not be configured by the user. If no timing configuration is provided by the user, a fairly reasonable timing behaviour will be used by default - one which results in fairly timely updates without placing an inordinate load on the server.

The following statements can be used to adjust the timing behaviour of the DHCP client if required, however:

The **timeout** *statement*

timeout *time* **;**

The *timeout* statement determines the amount of time that must pass between the time that the client begins to try to determine its address and the time that it decides that it's not going to be able to contact a server. By default, this timeout is sixty seconds. After the timeout has passed, if there are any static leases defined in the configuration file, or any leases remaining in the lease database that have not yet expired, the client will loop through these leases attempting to validate them, and if it finds one that appears to be valid, it will use that lease's address. If there are no valid static leases or unexpired leases in the lease database, the client will restart the protocol after the defined retry interval.

The **retry** *statement*

retry *time***;**

The *retry* statement determines the time that must pass after the client has determined that there is no DHCP server present before it tries again to contact a DHCP server. By default, this is five minutes.

The **select-timeout** *statement*

select-timeout *time***;**

It is possible (some might say desirable) for there to be more than one DHCP server serving any given network. In this case, it is possible that a client may be sent more than one offer in response to its initial lease discovery message. It may be that one of these offers is preferable to the other (e.g., one offer may have the address the client previously used, and the other may not).

The *select-timeout* is the time after the client sends its first lease discovery request at which it stops waiting for offers from servers, assuming that it has received at least one such offer. If no offers have been received by the time the *select-timeout* has expired, the client will accept the first offer that arrives.

By default, the select-timeout is zero seconds - that is, the client will take the first offer it sees.

The **reboot** *statement*

reboot *time*;

When the client is restarted, it first tries to reacquire the last address it had. This is called the INIT-REBOOT state. If it is still attached to the same network it was attached to when it last ran, this is the quickest way to get started. The *reboot* statement sets the time that must elapse after the client first tries to reacquire its old address before it gives up and tries to discover a new address. By default, the reboot timeout is ten seconds.

The **backoff-cutoff** *statement*

backoff-cutoff *time*;

The client uses an exponential backoff algorithm with some randomness, so that if many clients try to configure themselves at the same time, they will not make their requests in lockstep. The *backoff-cutoff* statement determines the maximum amount of time that the client is allowed to back off, the actual value will be evaluated randomly between 1/2 to 1 1/2 times the *time* specified. It defaults to fifteen seconds.

The **initial-interval** *statement*

initial-interval *time*;

The *initial-interval* statement sets the amount of time between the first attempt to reach a server and the second attempt to reach a server. Each time a message is sent, the interval between messages is incremented by twice the current interval multiplied by a random number between zero and one. If it is greater than the backoff-cutoff amount, it is set to that amount. It defaults to ten seconds.

The initial-delay statement

initial-delay *time*;

initial-delay parameter sets the maximum time client can wait after start before commencing first transmission. According to RFC2131 Section 4.4.1, client should wait a random time between startup and the actual first transmission. Previous versions of ISC DHCP client used to wait random time up to 5 seconds, but that was unwanted due to impact on startup time. As such, new versions have the default initial delay set to 0. To restore old behavior, please set initial-delay to 5.

LEASE REQUIREMENTS AND REQUESTS

The DHCP protocol allows the client to request that the server send it specific information, and not send it other information that it is not prepared to accept. The protocol also allows the client to reject offers from servers if they don't contain information the client needs, or if the information provided is not satisfactory.

There is a variety of data contained in offers that DHCP servers send to DHCP clients. The data that can be specifically requested is what are called *DHCP Options*. DHCP Options are defined in **dhcp-options(5)**.

The **request** *statement*

[**also**] **request** [[*option-space* .] *option*] [, ...];

The request statement causes the client to request that any server responding to the client send the client its values for the specified options. Only the option names should be specified in the request statement - not option parameters. By default, the DHCPv4 client requests the subnet-mask, broadcast-address, time-offset, routers, domain-name, domain-name-servers and host-name options while the DHCPv6 client requests the dhcp6 name-servers and domain-search options. Note that if you enter a ´request´ statement, you over-ride these defaults and these options will not be requested.

In some cases, it may be desirable to send no parameter request list at all. To do this, simply write the request statement but specify no parameters:

request;

In most cases, it is desirable to simply add one option to the request list which is of interest to the client in question. In this case, it is best to ´also request´ the additional options:

also request domain-search, dhcp6.sip-servers-addresses;

The **require** *statement*

[also] require [[*option-space* .] *option*] [**,** *...*]**;**

The require statement lists options that must be sent in order for an offer to be accepted. Offers that do not contain all the listed options will be ignored. There is no default require list.

require name-servers;

interface eth0 {
also require domain-search;
}

The
send
statement

send { [*option declaration*]
[**,** *... option declaration*]}

The send statement causes the client to send the specified options to the server with the specified values. These are full option declarations as described in **dhcp-options(5)**. Options that are always sent in the DHCP protocol should not be specified here, except that the client can specify a requested **dhcp-lease-time** option other than the default requested lease time, which is two hours. The other obvious use for this statement is to send information to the server that will allow it to differentiate between this client and other clients or kinds of clients.

DYNAMIC DNS

The client now has some very limited support for doing DNS updates when a lease is acquired. This is prototypical, and probably doesn't do what you want. It also only works if you happen to have control over your DNS server, which isn't very likely.

Note that everything in this section is true whether you are using DHCPv4 or DHCPv6. The exact same syntax is used for both.

To make it work, you have to declare a key and zone as in the DHCP server (see **dhcpd.conf**(5) for details). You also need to configure the *fqdn* option on the client, as follows:

send fqdn.fqdn grosse.fugue.com.;
send fqdn.encoded on;
send fqdn.server-update off;
also request fqdn, dhcp6.fqdn;

The *fqdn.fqdn* option **MUST** be a fully-qualified domain name. You **MUST** define a zone statement for the zone to be updated. The *fqdn.encoded* option may need to be set to *on* or *off*, depending on the DHCP server you are using.

The **do-forward-updates** *statement*

do-forward-updates [*flag*] **;**

If you want to do DNS updates in the DHCP client script (see **dhclient-script(8)**) rather than having the DHCP client do the update directly (for example, if you want to use SIG(0) authentication, which is not supported directly by the DHCP client, you can instruct the client not to do the update using the **do-forward-updates** statement. *Flag* should be **true** if you want the DHCP client to do the update, and **false** if you don't want the DHCP client to do the update. By

default, the DHCP client will do the DNS update.

OPTION MODIFIERS

In some cases, a client may receive option data from the server which is not really appropriate for that client, or may not receive information that it needs, and for which a useful default value exists. It may also receive information which is useful, but which needs to be supplemented with local information. To handle these needs, several option modifiers are available.

The **default** *statement*

default [*option declaration*] **;**

If for some option the client should use the value supplied by the server, but needs to use some default value if no value was supplied by the server, these values can be defined in the **default** statement.

The **supersede** *statement*

supersede [*option declaration*] **;**

If for some option the client should always use a locally-configured value or values rather than whatever is supplied by the server, these values can be defined in the **supersede** statement.

The **prepend** *statement*

prepend [*option declaration*] **;**

If for some set of options the client should use a value you supply, and then use the values supplied by the server, if any, these values can be defined in the **prepend** statement. The **prepend** statement can only be used for options which allow more than one value to be given. This restriction is not enforced - if you ignore it, the behaviour will be unpredictable.

The **append** *statement*

append [*option declaration*] **;**

If for some set of options the client should first use the values supplied by the server, if any, and then use values you supply, these values can be defined in the **append** statement. The **append** statement can only be used for options which allow more than one value to be given. This restriction is not enforced - if you ignore it, the behaviour will be unpredictable.

LEASE DECLARATIONS

The **lease** *declaration*

lease { *lease-declaration* [*... lease-declaration]* }

The DHCP client may decide after some period of time (see **PROTOCOL TIMING**) that it is not going to succeed in contacting a server. At that time, it consults its own database of old leases and tests each one that has not yet timed out by pinging the listed router for that lease to see if that lease could work. It is possible to define one or more *fixed* leases in the client configuration file for networks where there is no DHCP or BOOTP service, so that the client can still automatically configure its address. This is done with the **lease** statement.

NOTE: the lease statement is also used in the dhclient.leases file in order to record leases that have been received from DHCP servers. Some of the syntax for leases as described below is only needed in the dhclient.leases file. Such syntax is documented here for completeness.

A lease statement consists of the lease keyword, followed by a left curly brace, followed by one or more lease declaration statements, followed by a right curly brace. The following lease declarations are possible:

bootp;

The **bootp** statement is used to indicate that the lease was acquired using the BOOTP protocol rather than the DHCP protocol. It is never necessary to specify this in the client configuration file. The client uses this syntax in its lease database file.

interface *string*;

The **interface** lease statement is used to indicate the interface on which the lease is valid. If set, this lease will only be tried on a particular interface. When the client receives a lease from a server, it always records the interface number on which it received that lease. If predefined leases are specified in the dhclient.conf file, the interface should also be specified, although this is not required.

fixed-address *ip-address*;

The **fixed-address** statement is used to set the ip address of a particular lease. This is required for all lease statements. The IP address must be specified as a dotted quad (e.g., 12.34.56.78).

filename *string*;

The **filename** statement specifies the name of the boot filename to use. This is not used by the standard client configuration script, but is included for completeness.

server-name *string*;

The **server-name** statement specifies the name of the boot server name to use. This is also not used by the standard client configuration script.

option *option-declaration*;

The **option** statement is used to specify the value of an option supplied by the server, or, in the case of predefined leases declared in dhclient.conf, the value that the user wishes the client configuration script to use if the predefined lease is used.

script *script-name*;

The **script** statement is used to specify the pathname of the dhcp client configuration script. This script is used by the dhcp client to set each interface's initial configuration prior to requesting an address, to test the address once it has been offered, and to set the interface's final configuration once a lease has been acquired. If no lease is acquired, the script is used to test predefined leases, if any, and also called once if no valid lease can be identified. For more information, see dhclient-script(8)

vendor option space *name*;

The **vendor option space** statement is used to specify which option space should be used for decoding the vendor-encapsulate-options option if one is received. The*dhcp-vendor-identifier* can be used to request a specific class of vendor options from the server. See dhcp-options(5) for details.

medium *media setup*;

The **medium** statement can be used on systems where network interfaces cannot automatically determine the type of network to which they are connected. The media setup string is a system-dependent parameter which is passed to the dhcp client configuration script when initializing the interface. On Unix and Unix-like systems, the argument is passed on the ifconfig command line when configuring the interface.

The dhcp client automatically declares this parameter if it uses a media type (see the **media** statement) when configuring the interface in order to obtain a lease. This statement should be used in predefined leases only if the network interface requires media type configuration.

renew *date*;

rebind *date*;

expire *date*;

The **renew** statement defines the time at which the dhcp client should begin trying to contact its server to renew a lease that it is using. The **rebind** statement defines the time at which the dhcp client should begin to try to contact *any* dhcp server in order to renew its lease. The **expire**

statement defines the time at which the dhcp client must stop using a lease if it has not been able to contact a server in order to renew it.

These declarations are automatically set in leases acquired by the DHCP client, but must also be configured in predefined leases - a predefined lease whose expiry time has passed will not be used by the DHCP client.

Dates are specified in one of two ways. The software will output times in these two formats depending on if the **db-time-format** configuration parameter has been set to *default* or *local*.

If it is set to *default*, then *date* values appear as follows:

<*weekday*> <*year*>/<*month*>/<*day*> <*hour*>:<*minute*>:<*second*>

The weekday is present to make it easy for a human to tell when a lease expires - it's specified as a number from zero to six, with zero being Sunday. When declaring a predefined lease, it can always be specified as zero. The year is specified with the century, so it should generally be four digits except for really long leases. The month is specified as a number starting with 1 for January. The day of the month is likewise specified starting with 1. The hour is a number between 0 and 23, the minute a number between 0 and 59, and the second also a number between 0 and 59.

If the **db-time-format** configuration was set to *local*, then the *date* values appear as follows:

epoch <*seconds-since-epoch*>**;** #<*day-name*> <*month-name*> <*day-numb er*> <*hours*>:<*minutes*>:<*seconds*> <*year*>

The *seconds-since-epoch* is as according to the system's local clock (often referred to as unix time). The # symbol supplies a comment that describes what actual time this is as according to the system's configured timezone, at the time the value was written. It is provided only for human inspection, the epoch time is the only recommended value for machine inspection.

Note that when defining a static lease, one may use either time format one wishes, and need not include the comment or values after it.

If the time is infinite in duration, then the *date* is **never** instead of an actual date.

ALIAS DECLARATIONS

alias { *declarations ...* }

Some DHCP clients running TCP/IP roaming protocols may require that in addition to the lease they may acquire via DHCP, their interface also be configured with a predefined IP alias so that they can have a permanent IP address even while roaming. The Internet Systems Consortium DHCP client doesn't support roaming with fixed addresses directly, but in order to facilitate such experimentation, the dhcp client can be set up to configure an IP alias using the **alias** declaration.

The alias declaration resembles a lease declaration, except that options other than the subnet-mask option are ignored by the standard client configuration script, and expiry times are ignored. A typical alias declaration includes an interface declaration, a fixed-address declaration for the IP alias address, and a subnet-mask option declaration. A medium statement should never be included in an alias declaration.

OTHER DECLARATIONS

db-time-format [*default* | *local*] **;**

The **db-time-format** option determines which of two output methods are used for printing times in leases files. The *default* format provides day-and-time in UTC, whereas *local* uses a seconds-since-epoch to store the time value, and helpfully places a local timezone time in a comment on the same line. The formats are described in detail in this manpage, within the LEASE DECLARATIONS section.

reject *cidr-ip-address* [**,** *... cidr-ip-address*] **;**

The **reject** statement causes the DHCP client to reject offers from servers whose server identifier matches any of the specified hosts or subnets. This can be used to avoid being configured by

rogue or misconfigured dhcp servers, although it should be a last resort - better to track down the bad DHCP server and fix it.

The *cidr-ip-address* configuration type is of the form *ip-address*[/*prefixlen*], where *ip-address* is a dotted quad IP address, and prefixlen is the CIDR prefix length of the subnet, counting the number of significant bits in the netmask starting from the leftmost end. Example configuration syntax:

reject 192.168.0.0/16**,** 10.0.0.5**;**

The above example would cause offers from any server identifier in the entire RFC 1918 Class C network 192.168.0.0/16, or the specific single address 10.0.0.5, to be rejected.

interface *name* { *declarations ...* }

A client with more than one network interface may require different behaviour depending on which interface is being configured. All timing parameters and declarations other than lease and alias declarations can be enclosed in an interface declaration, and those parameters will then be used only for the interface that matches the specified name. Interfaces for which there is no interface declaration will use the parameters declared outside of any interface declaration, or the default settings.

Note well: ISC dhclient only maintains one list of interfaces, which is either determined at startup from command line arguments, or otherwise is autodetected. If you supplied the list of interfaces on the command line, this configuration clause will add the named interface to the list in such a way that will cause it to be configured by DHCP. Which may not be the result you had intended. This is an undesirable side effect that will be addressed in a future release.

pseudo *name real-name* { *declarations ...* }

Under some circumstances it can be useful to declare a pseudo-interface and have the DHCP client acquire a configuration for that interface. Each interface that the DHCP client is supporting normally has a DHCP client state machine running on it to acquire and maintain its lease. A pseudo-interface is just another state machine running on the interface named *real-name*, with its own lease and its own state. If you use this feature, you must provide a client identifier for both the pseudo-interface and the actual interface, and the two identifiers must be different. You must also provide a separate client script for the pseudo-interface to do what you want with the IP address. For example:

```
interface ep0 {
send dhcp-client-identifier my-client-ep0;
}
pseudo secondary ep0 {
send dhcp-client-identifier my-client-ep0-secondary;
script /etc/dhclient-secondary;
}
```

The client script for the pseudo-interface should not configure the interface up or down - essentially, all it needs to handle are the states where a lease has been acquired or renewed, and the states where a lease has expired. See **dhclient-script(8)** for more information.

media *media setup [* **,** *media setup***,** *...]***;**

The **media** statement defines one or more media configuration parameters which may be tried while attempting to acquire an IP address. The dhcp client will cycle through each media setup string on the list, configuring the interface using that setup and attempting to boot, and then trying the next one. This can be used for network interfaces which aren't capable of sensing the media type unaided - whichever media type succeeds in getting a request to the server and hearing the reply is probably right (no guarantees).

The media setup is only used for the initial phase of address acquisition (the DHCPDISCOVER and DHCPOFFER packets). Once an address has been acquired, the dhcp client will record it in

its lease database and will record the media type used to acquire the address. Whenever the client tries to renew the lease, it will use that same media type. The lease must expire before the client will go back to cycling through media types.

hardware *link-type mac-address***;**

The **hardware** statement defines the hardware MAC address to use for this interface, for DHCP servers or relays to direct their replies. dhclient will determine the interface's MAC address automatically, so use of this parameter is not recommended. The *link-type* corresponds to the interface's link layer type (example: ´ethernet´), while the *mac-address* is a string of colon-separated hexadecimal values for octets.

anycast-mac *link-type mac-address***;**

The **anycast-mac** statement over-rides the all-ones broadcast MAC address dhclient will use when it is transmitting packets to the all-ones limited broadcast IPv4 address. This configuration parameter is useful to reduce the number of broadcast packets transmitted by DHCP clients, but is only useful if you know the DHCP service(s) anycast MAC address prior to configuring your client. The *link-type* and *mac-address* parameters are configured in a similar manner to the **hardware** statement.

SAMPLE

The following configuration file is used on a laptop running NetBSD 1.3. The laptop has an IP alias of 192.5.5.213, and has one interface, ep0 (a 3com 3C589C). Booting intervals have been shortened somewhat from the default, because the client is known to spend most of its time on networks with little DHCP activity. The laptop does roam to multiple networks.

```
timeout 60;
retry 60;
reboot 10;
select-timeout 5;
initial-interval 2;
reject 192.33.137.209;

interface ep0 {
send host-name andare.fugue.com;
hardware ethernet 00:a0:24:ab:fb:9c;
send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
send dhcp-lease-time 3600;
supersede domain-search fugue.com, rc.vix.com, home.vix.com;
prepend domain-name-servers 127.0.0.1;
request subnet-mask, broadcast-address, time-offset, routers,
domain-name, domain-name-servers, host-name;
require subnet-mask, domain-name-servers;
script CLIENTBINDIR/dhclient-script;
media media 10baseT/UTP, media 10base2/BNC;
}

alias {
interface ep0;
fixed-address 192.5.5.213;
option subnet-mask 255.255.255.255;
}
```

This is a very complicated dhclient.conf file - in general, yours should be much simpler. In many cases, it's sufficient to just create an empty dhclient.conf file - the defaults are usually fine.

SEE ALSO

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

AUTHOR

dhclient(8) Information about Internet Systems Consortium can be found at https://www.isc.org.