NAME

dhclient - Dynamic Host Configuration Protocol Client

SYNOPSIS

dhclient [**-4** | **-6**] [**-S**] [**-N** [**-N...**]] [**-T** [**-T...**]] [**-P** [**-P...**]] **-R**] [**-i**] [**-I**] [**-4o6** *port*] [**-D** *LL|LLT*] [**-p** *port-number*] [**-d**] [**-df** *duid-lease-file*] [**-e** *VAR=value*] [**-q**] [**-1**] [**-r** | **-x**] [**-lf** *lease-file*] [**-pf** *pid-file*] [**--no-pid**] [**-cf** *config-file*] [**-sf** *script-file*] [**-s** *server-addr*] [**-g** *relay*] [**-n**] [**-nw**] [**-w**] [**--dad-wait-time** *seconds*] [**--prefix-len-hint** *length*] [**--decline-wait-time** *seconds*] [**-v**] [**--version**] [*if0* [*...ifN*]]

DESCRIPTION

The Internet Systems Consortium DHCP Client, **dhclient**, provides a means for configuring one or more network interfaces using the Dynamic Host Configuration Protocol, BOOTP protocol, or if these protocols fail, by statically assigning an address.

OPERATION

The DHCP protocol allows a host to contact a central server which maintains a list of IP addresses which may be assigned on one or more subnets. A DHCP client may request an address from this pool, and then use it on a temporary basis for communication on network. The DHCP protocol also provides a mechanism whereby a client can learn important details about the network to which it is attached, such as the location of a default router, the location of a name server, and so on.

There are two versions of the DHCP protocol DHCPv4 and DHCPv6. At startup the client may be started for one or the other via the **-4** or **-6** options.

On startup, **dhclient** reads the dhclient.conf for configuration instructions. It then gets a list of all the network interfaces that are configured in the current system. For each interface, it attempts to configure the interface using the DHCP protocol.

In order to keep track of leases across system reboots and server restarts, **dhclient** keeps a list of leases it has been assigned in the dhclient.leases file. On startup, after reading the dhclient.conf file, **dhclient** reads the dhclient.leases file to refresh its memory about what leases it has been assigned.

When a new lease is acquired, it is appended to the end of the dhclient.leases file. In order to prevent the file from becoming arbitrarily large, from time to time **dhclient** creates a new dhclient.leases file from its in-core lease database. The old version of the dhclient.leases file is retained under the name *dhclient.leases~* until the next time **dhclient** rewrites the database.

Old leases are kept around in case the DHCP server is unavailable when **dhclient** is first invoked (generally during the initial system boot process). In that event, old leases from the dhclient.leases file which have not yet expired are tested, and if they are determined to be valid, they are used until either they expire or the DHCP server becomes available.

A mobile host which may sometimes need to access a network on which no DHCP server exists may be preloaded with a lease for a fixed address on that network. When all attempts to contact a DHCP server have failed, **dhclient** will try to validate the static lease, and if it succeeds, will use that lease until it is restarted.

A mobile host may also travel to some networks on which DHCP is not available but BOOTP is. In that case, it may be advantageous to arrange with the network administrator for an entry on the BOOTP database, so that the host can boot quickly on that network rather than cycling through the list of old leases.

COMMAND LINE

The names of the network interfaces that **dhclient** should attempt to configure may be specified on the command line. If no interface names are specified on the command line **dhclient** will normally identify all network interfaces, eliminating non-broadcast interfaces if possible, and attempt to configure each interface.

It is also possible to specify interfaces by name in the dhclient.conf file. If interfaces are specified in this way, then the client will only configure interfaces that are either specified in the configuration file or on the command line, and will ignore all other interfaces.

The client normally prints no output during its startup sequence. It can be made to emit verbose messages

displaying the startup sequence events until it has acquired an address by supplying the **-v** command line argument. In either case, the client logs messages using the **syslog(3)** facility.

OPTIONS

–4 Use the DHCPv4 protocol to obtain an IPv4 address and configuration parameters. This is the default and cannot be combined with **–6**.

–6 Use the DHCPv6 protocol to obtain whatever IPv6 addresses are available along with configuration parameters. It cannot be combined with **–4**. The **–S -T -P -N** and **–D** arguments provide more control over aspects of the DHCPv6 processing. Note: it is not recommended to mix queries of different types together or even to share the lease file between them.

–4o6 *port*
Participate in the DHCPv4 over DHCPv6 protocol specified by RFC 7341. This associates a DHCPv4 and a DHCPv6 client to allow the v4 client to send v4 requests encapsulated in a v6 packet. Communication between the two clients is done on a pair of UDP sockets bound to ::1 *port* and *port + 1*. Both clients must be launched using the same *port* argument.

–1 Try to get a lease once. On failure exit with code 2. In DHCPv6 this sets the maximum duration of the initial exchange to *timeout* (from dhclient.conf with a default of sixty seconds).

–d Force **dhclient** to run as a foreground process. Normally the DHCP client will run in the foreground until is has configured an interface at which time it will revert to running in the background. This option is useful when running the client under a debugger, or when running it out of inittab on System V systems. This implies **-v**.

–nw Become a daemon immediately (nowait) rather than waiting until an IP address has been acquired.

–q Be quiet at startup, this is the default.

–v Enable verbose log messages.

–w Continue running even if no broadcast interfaces were found. Normally DHCP client will exit if it isn't able to identify any network interfaces to configure. On laptop computers and other computers with hot-swappable I/O buses, it is possible that a broadcast interface may be added after system startup. This flag can be used to cause the client not to exit when it doesn't find any such interfaces. The **omshell(1)** program can then be used to notify the client when a network interface has been added or removed, so that the client can attempt to configure an IP address on that interface.

–n Do not configure any interfaces. This is most likely to be useful in combination with the **-w** flag.

–e *VAR=value*
Define additional environment variables for the environment where **dhclient-script** executes. You may specify multiple **–e** options on the command line.

–r Release the current lease and stop the running DHCP client as previously recorded in the PID file. When shutdown via this method **dhclient-script** will be executed with the specific reason for calling the script set. The client normally doesn't release the current lease as this is not required by the DHCP protocol but some cable ISPs require their clients to notify the server if they wish to release an assigned IP address.

–x Stop the running DHCP client without releasing the current lease. Kills existing **dhclient** process as previously recorded in the PID file. When shutdown via this method **dhclient-script** will be executed with the specific reason for calling the script set.

–p *port-number*
The UDP port number on which the DHCP client should listen and transmit. If unspecified, **dhclient** uses the default port of 68. This is mostly useful for debugging purposes. If a different port is specified on which the client should listen and transmit, the client will also use a different destination port - one less than the specified port.

–s *server-addr*
Specify the server IP address or fully qualified domain name to use as a destination for DHCP protocol messages before **dhclient** has acquired an IP address. Normally, **dhclient** transmits these messages to 255.255.255.255 (the IP limited broadcast address). Overriding this is mostly useful for debugging purposes. This feature is not supported in DHCPv6 (**-6**) mode.

–g *relay*
Set the giaddr field of all packets to the *relay* IP address simulating a relay agent. This is for testing purposes only and should not be expected to work in any consistent or useful way.

–i Use a DUID with DHCPv4 clients. If no DUID is available in the lease file one will be constructed and saved. The DUID will be used to construct a RFC4361 style client id that will be included in the client's messages. This client id can be overridden by setting a client id in the configuration file. Overriding the client id in this fashion is discouraged.

–I Use the standard DDNS scheme from RFCs 4701 & 4702.

–-decline-wait-time *seconds*
Specify the time (in seconds) that an IPv4 client should wait after declining an address before issuing a discover. The default is 10 seconds as recommended by RFC 2131, Section 3.1.5. A value of zero equates to no wait at all.

–-version Print version number and exit.

Options available for DHCPv6 mode:

–S Use Information-request to get only stateless configuration parameters (i.e., without address). This implies **–6**. It also doesn't rewrite the lease database.

–T Ask for IPv6 temporary addresses, one set per **–T** flag. This implies **–6** and also disables the normal address query. See **–N** to restore it.

–P Enable IPv6 prefix delegation. This implies **–6** and also disables the normal address query. See **–N** to restore it. Multiple prefixes can be requested with multiple **–P** flags. Note only one requested interface is allowed.

–R Require that responses include all of the items requested by any **–N**, **–T**, or **–P** options. Normally even if the command line includes a number of these the client will be willing to accept the best lease it can even if the lease doesn't include all of the requested items. This option causes the client to only accept leases that include all of the requested items.

Note well: enabling this may prevent the client from using any leases it receives if the servers aren't configured to supply all of the items.

–D *LL or LLT*
Override the default when selecting the type of DUID to use. By default, DHCPv6 **dhclient** creates an identifier based on the link-layer address (DUID-LL) if it is running in stateless mode (with **–S**, not requesting an address), or it creates an identifier based on the link-layer address plus a timestamp (DUID-LLT) if it is running in stateful mode (without **–S**, requesting an address). When DHCPv4 is configured to use a DUID using **–i** option the default is to use a DUID-LLT. **–D** overrides these default, with a value of either *LL* or *LLT*.

–N Restore normal address query for IPv6. This implies **-6**. It is used to restore normal operation after using **-T** or **-P**. Multiple addresses can be requested with multiple **–N** flags.

–-address-prefix-len *length*
Specify the length of the prefix for IPv6 addresses. This value is passed by dhclient into the client script via the environment variable, ip6_prefixlen, when binding IPv6 addresses. The default value is 128. Alternatively you may change the default at compile time by setting DHCLIENT_DEFAULT_PREFIX_LEN in includes/site.h.

–-dad-wait-time *seconds*
Specify maximum time (in seconds) that the client should wait for the duplicate address detection (DAD) to complete on an interface. This value is propagated to the dhclient script in a dad_wait_time environment variable. If any of the IPv6 addresses on the interface are tentative (DAD is in progress), the script will wait for the specified number of seconds for DAD to complete. If the script ignores this variable the parameter has no effect.

–-prefix-len-hint *length*
When used in conjunction with -P, it directs the client to use the given length to use a prefix hint of, "::/length", when requesting new prefixes.

Modifying default file locations: The following options can be used to modify the locations a client uses for its files. They can be particularly useful if, for example, **/var/lib/dhcp** or **/var/run** have not been mounted when the DHCP client is started.

–cf *config-file*
Path to the client configuration file. If unspecified, the default **/etc/dhcp/dhclient.conf** is used. See **dhclient.conf(5)** for a description of this file.

–df *duid-lease-file*
Path to a secondary lease file. If the primary lease file doesn't contain a DUID this file will be searched. The DUID read from the secondary will be written to the primary. This option can be used to allow an IPv4 instance of the client to share a DUID with an IPv6 instance. After starting one of the instances the second can be started with this option pointing to the lease file of the first instance. There is no default. If no file is specified no search is made for a DUID should one not be found in the main lease file.

–lf *lease-file*
Path to the lease database file. If unspecified, the default **/var/lib/dhcp/dhclient.leases** is used. See **dhclient.leases(5)** for a description of this file.

–pf *pid-file*
Path to the process ID file. If unspecified, the default **/var/run/dhclient.pid** is used.

–-no-pid
Option to disable writing pid files. By default the program will write a pid file. If the program is invoked with this option it will not attempt to kill any existing client processes even if invoked with **-r** or **-x**.

–sf *script-file*
Path to the network configuration script invoked by **dhclient** when it gets a lease. If unspecified, the default **/sbin/dhclient-script** is used. See **dhclient-script(8)** for a description of this file.

PORTS

During operations the client may use multiple UDP ports to provide different functions. Which ports are opened depends on both the way you compiled your code and the configuration you supply. The following should provide you an idea of what ports may be in use.

Normally a DHCPv4 client will open a raw UDP socket to receive and send most DHCPv4 packets. It also opens a fallback UDP socket for use in sending unicast packets. Normally these will both use the well known port number for BOOTPC.

For DHCPv6 the client opens a UDP socket on the well known client port and a fallback UDP socket on a random port for use in sending unicast messages. Unlike DHCPv4 the well known socket doesn't need to be opened in raw mode.

If you have included an omapi port statement in your configuration file then the client will open a TCP socket on that port to listen for OMPAI connections. When something connects another port will be used for the established connection.

When DDNS is enabled at compile time (see includes/site.h) the client will open both a v4 and a v6 UDP socket on random ports. These ports are not opened unless/until the client first attempts to do an update. If the client is not configured to do updates, the ports will never be opened.

CONFIGURATION

The syntax of the **dhclient.conf(5)** file is discussed separately.

OMAPI

The DHCP client provides some ability to control it while it is running, without stopping it. This capability is provided using OMAPI, an API for manipulating remote objects. OMAPI clients connect to the client using TCP/IP, authenticate, and can then examine the client's current status and make changes to it.

Rather than implementing the underlying OMAPI protocol directly, user programs should use the dhcpctl API or OMAPI itself. Dhcpctl is a wrapper that handles some of the housekeeping chores that OMAPI does not do automatically. Dhcpctl and OMAPI are documented in **dhcpctl(3)** and **omapi(3)**. Most things you'd want to do with the client can be done directly using the **omshell(1)** command, rather than having to write a special program.

THE CONTROL OBJECT

The control object allows you to shut the client down, releasing all leases that it holds and deleting any DNS records it may have added. It also allows you to pause the client - this unconfigures any interfaces the client is using. You can then restart it, which causes it to reconfigure those interfaces. You would normally pause the client prior to going into hibernation or sleep on a laptop computer. You would then resume it after the power comes back. This allows PC cards to be shut down while the computer is hibernating or sleeping, and then reinitialized to their previous state once the computer comes out of hibernation or sleep.

The control object has one attribute - the state attribute. To shut the client down, set its state attribute to 2. It will automatically do a DHCPRELEASE. To pause it, set its state attribute to 3. To resume it, set its state attribute to 4.

ENVIRONMENT VARIABLES

The following environment variables may be defined to override the builtin defaults for file locations. Note that use of the related command-line options will ignore the corresponding environment variable settings.

PATH_DHCLIENT_CONF
The dhclient.conf configuration file.

PATH_DHCLIENT_DB
The dhclient.leases database.

PATH_DHCLIENT_PID
The dhclient PID file.

PATH_DHCLIENT_SCRIPT
The dhclient-script file.

FILES

/sbin/dhclient-script, /etc/dhcp/dhclient.conf, /var/lib/dhcp/dhclient.leases, /var/run/dhclient.pid, /var/lib/dhcp/dhclient.leases~.

SEE ALSO

dhcpd(8), dhcrelay(8), dhclient-script(8), dhclient.conf(5), dhclient.leases(5), dhcp-eval(5).

AUTHOR

dhclient(8) To learn more about Internet Systems Consortium, see **https://www.isc.org**

This client was substantially modified and enhanced by Elliot Poger for use on Linux while he was working on the MosquitoNet project at Stanford.

The current version owes much to Elliot's Linux enhancements, but was substantially reorganized and partially rewritten by Ted Lemon so as to use the same networking framework that the Internet Systems Consortium DHCP server uses. Much system-specific configuration code was moved into a shell script so that as support for more operating systems is added, it will not be necessary to port and maintain system-

specific configuration code to these operating systems - instead, the shell script can invoke the native tools to accomplish the same purpose.