NAME

dnsmasq − A lightweight DHCP and caching DNS server.

SYNOPSIS

dnsmasq *[OPTION]...*

DESCRIPTION

dnsmasq is a lightweight DNS, TFTP, PXE, router advertisement and DHCP server. It is intended to provide coupled DNS and DHCP service to a LAN.

Dnsmasq accepts DNS queries and either answers them from a small, local, cache or forwards them to a real, recursive, DNS server. It loads the contents of /etc/hosts so that local hostnames which do not appear in the global DNS can be resolved and also answers DNS queries for DHCP configured hosts. It can also act as the authoritative DNS server for one or more domains, allowing local names to appear in the global DNS. It can be configured to do DNSSEC validation.

The dnsmasq DHCP server supports static address assignments and multiple networks. It automatically sends a sensible default set of DHCP options, and can be configured to send any desired set of DHCP options, including vendor-encapsulated options. It includes a secure, read-only, TFTP server to allow net/PXE boot of DHCP hosts and also supports BOOTP. The PXE support is full featured, and includes a proxy mode which supplies PXE information to clients whilst DHCP address allocation is done by another server.

The dnsmasq DHCPv6 server provides the same set of features as the DHCPv4 server, and in addition, it includes router advertisements and a neat feature which allows naming for clients which use DHCPv4 and stateless autoconfiguration only for IPv6 configuration. There is support for doing address allocation (both DHCPv6 and RA) from subnets which are dynamically delegated via DHCPv6 prefix delegation.

Dnsmasq is coded with small embedded systems in mind. It aims for the smallest possible memory footprint compatible with the supported functions, and allows unneeded functions to be omitted from the compiled binary.

OPTIONS

Note that in general missing parameters are allowed and switch off functions, for instance "--pid-file" disables writing a PID file. On BSD, unless the GNU getopt library is linked, the long form of the options does not work on the command line; it is still recognised in the configuration file.

--test Read and syntax check configuration file(s). Exit with code 0 if all is OK, or a non-zero code otherwise. Do not start up dnsmasq.

−w, --help
Display all command-line options. **--help dhcp** will display kno wn DHCPv4 configuration options, and **--help dhcp6** will display DHCPv6 options.

−h, --no-hosts
Don't read the hostnames in /etc/hosts.

−H, --addn-hosts=<file>
Additional hosts file. Read the specified file as well as /etc/hosts. If **--no-hosts** is given, read only the specified file. This option may be repeated for more than one additional hosts file. If a directory is given, then read all the files contained in that directory in alphabetical order.

--hostsdir=<path>
Read all the hosts files contained in the directory. New or changed files are read automatically. See **--dhcp-hostsdir** for details.

−E, --expand-hosts
Add the domain to simple names (without a period) in /etc/hosts in the same way as for DHCP-derived names. Note that this does not apply to domain names in cnames, PTR records, TXT records etc.

−T, --local-ttl=<time>
When replying with information from /etc/hosts or configuration or the DHCP leases file dnsmasq by default sets the time-to-live field to zero, meaning that the requester should not itself cache the

information. This is the correct thing to do in almost all situations. This option allows a time-to-live (in seconds) to be given for these replies. This will reduce the load on the server at the expense of clients using stale data under some circumstances.

--dhcp-ttl=<time>
As for **--local-ttl**, but affects only replies with information from DHCP leases. If both are given, **--dhcp-ttl** applies for DHCP information, and **--local-ttl** for others. Setting this to zero eliminates the effect of **--local-ttl** for DHCP.

--neg-ttl=<time>
Negative replies from upstream servers normally contain time-to-live information in SOA records which dnsmasq uses for caching. If the replies from upstream servers omit this information, dnsmasq does not cache the reply. This option gives a default value for time-to-live (in seconds) which dnsmasq uses to cache negative replies even in the absence of an SOA record.

--max-ttl=<time>
Set a maximum TTL value that will be handed out to clients. The specified maximum TTL will be given to clients instead of the true TTL value if it is lower. The true TTL value is however kept in the cache to avoid flooding the upstream DNS servers.

--max-cache-ttl=<time>
Set a maximum TTL value for entries in the cache.

--min-cache-ttl=<time>
Extend short TTL values to the time given when caching them. Note that artificially extending TTL values is in general a bad idea, do not do it unless you have a good reason, and understand what you are doing. Dnsmasq limits the value of this option to one hour, unless recompiled.

--auth-ttl=<time>
Set the TTL value returned in answers from the authoritative server.

−k, --keep-in-foreground
Do not go into the background at startup but otherwise run as normal. This is intended for use when dnsmasq is run under daemontools or launchd.

−d, --no-daemon
Debug mode: don't fork to the background, don't write a pid file, don't change user id, generate a complete cache dump on receipt on SIGUSR1, log to stderr as well as syslog, don't fork new processes to handle TCP queries. Note that this option is for use in debugging only, to stop dnsmasq daemonising in production, use **--keep-in-foreground.**

−q, --log-queries
Log the results of DNS queries handled by dnsmasq. Enable a full cache dump on receipt of SIGUSR1. If the argument "extra" is supplied, ie **--log-queries=extra** then the log has extra information at the start of each line. This consists of a serial number which ties together the log lines associated with an individual query, and the IP address of the requestor.

−8, --log-facility=<facility>
Set the facility to which dnsmasq will send syslog entries, this defaults to DAEMON, and to LOCAL0 when debug mode is in operation. If the facility given contains at least one '/' character, it is taken to be a filename, and dnsmasq logs to the given file, instead of syslog. If the facility is '-' then dnsmasq logs to stderr. (Errors whilst reading configuration will still go to syslog, but all output from a successful startup, and all output whilst running, will go exclusively to the file.) When logging to a file, dnsmasq will close and reopen the file when it receives SIGUSR2. This allows the log file to be rotated without stopping dnsmasq.

--log-debug
Enable extra logging intended for debugging rather than information.

--log-async[=<lines>]
Enable asynchronous logging and optionally set the limit on the number of lines which will be queued by dnsmasq when writing to the syslog is slow. Dnsmasq can log asynchronously: this

allows it to continue functioning without being blocked by syslog, and allows syslog to use dnsmasq for DNS queries without risking deadlock. If the queue of log-lines becomes full, dnsmasq will log the overflow, and the number of messages lost. The default queue length is 5, a sane value would be 5-25, and a maximum limit of 100 is imposed.

−x, --pid-file=<path>
Specify an alternate path for dnsmasq to record its process-id in. Normally /var/run/dnsmasq.pid.

−u, --user=<username>
Specify the userid to which dnsmasq will change after startup. Dnsmasq must normally be started as root, but it will drop root privileges after startup by changing id to another user. Normally this user is "nobody" but that can be over-ridden with this switch.

−g, --group=<groupname>
Specify the group which dnsmasq will run as. The default is "dip", if available, to facilitate access to /etc/ppp/resolv.conf which is not normally world readable.

−v, --version
Print the version number.

−p, --port=<port>
Listen on <port> instead of the standard DNS port (53). Setting this to zero completely disables DNS function, leaving only DHCP and/or TFTP.

−P, --edns-packet-max=<size>
Specify the largest EDNS.0 UDP packet which is supported by the DNS forwarder. Defaults to 4096, which is the RFC5625-recommended size.

−Q, --query-port=<query_port>
Send outbound DNS queries from, and listen for their replies on, the specific UDP port <query_port> instead of using random ports. NOTE that using this option will make dnsmasq less secure against DNS spoofing attacks but it may be faster and use less resources. Setting this option to zero makes dnsmasq use a single port allocated to it by the OS: this was the default behaviour in versions prior to 2.43.

--min-port=<port>
Do not use ports less than that given as source for outbound DNS queries. Dnsmasq picks random ports as source for outbound queries: when this option is given, the ports used will always be larger than that specified. Useful for systems behind firewalls. If not specified, defaults to 1024.

--max-port=<port>
Use ports lower than that given as source for outbound DNS queries. Dnsmasq picks random ports as source for outbound queries: when this option is given, the ports used will always be lower than that specified. Useful for systems behind firewalls.

−i, --interface=<interface name>
Listen only on the specified interface(s). Dnsmasq automatically adds the loopback (local) interface to the list of interfaces to use when the **−-interface** option is used. If no **−-interface** or **−-listen-address** options are given dnsmasq listens on all available interfaces except any given in **−-except-interface** options. On Linux, when **−-bind-interfaces** or **−-bind-dynamic** are in effect, IP alias interface labels (eg "eth1:0") are checked, rather than interface names. In the degenerate case when an interface has one address, this amounts to the same thing but when an interface has multiple addresses it allows control over which of those addresses are accepted. The same effect is achievable in default mode by using **−-listen-address.** A simple wildcard, consisting of a trailing '*', can be used in **−-interface** and **−-except-interface** options.

−I, --except-interface=<interface name>
Do not listen on the specified interface. Note that the order of **−-listen-address --interface** and **--except-interface** options does not matter and that **--except-interface** options always override the others. The comments about interface labels for **--listen-address** apply here.

--auth-server=<domain>,[<interface>|<ip-address>...]
Enable DNS authoritative mode for queries arriving at an interface or address. Note that the interface or address need not be mentioned in **--interface** or **--listen-address** configuration, indeed **--auth-server** will override these and provide a different DNS service on the specified interface. The <domain> is the "glue record". It should resolve in the global DNS to an A and/or AAAA record which points to the address dnsmasq is listening on. When an interface is specified, it may be qualified with "/4" or "/6" to specify only the IPv4 or IPv6 addresses associated with the interface. Since any defined authoritative zones are also available as part of the normal recusive DNS service supplied by dnsmasq, it can make sense to have an --auth-server declaration with no interfaces or address, but simply specifying the primary external nameserver.

--local-service
Accept DNS queries only from hosts whose address is on a local subnet, ie a subnet for which an interface exists on the server. This option only has effect if there are no **--interface**, **--except-interface**, **--listen-address** or **--auth-server** options. It is intended to be set as a default on installation, to allow unconfigured installations to be useful but also safe from being used for DNS amplification attacks.

−2, --no-dhcp-interface=<interface name>
Do not provide DHCP or TFTP on the specified interface, but do provide DNS service.

−a, --listen-address=<ipaddr>
Listen on the given IP address(es). Both **−-interface** and **−-listen-address** options may be given, in which case the set of both interfaces and addresses is used. Note that if no **−-interface** option is given, but **−-listen-address** is, dnsmasq will not automatically listen on the loopback interface. To achieve this, its IP address, 127.0.0.1, must be explicitly given as a **−-listen-address** option.

−z, --bind-interfaces
On systems which support it, dnsmasq binds the wildcard address, even when it is listening on only some interfaces. It then discards requests that it shouldn't reply to. This has the advantage of working even when interfaces come and go and change address. This option forces dnsmasq to really bind only the interfaces it is listening on. About the only time when this is useful is when running another nameserver (or another instance of dnsmasq) on the same machine. Setting this option also enables multiple instances of dnsmasq which provide DHCP service to run in the same machine.

--bind-dynamic
Enable a network mode which is a hybrid between **--bind-interfaces** and the default. Dnsmasq binds the address of individual interfaces, allowing multiple dnsmasq instances, but if new interfaces or addresses appear, it automatically listens on those (subject to any access-control configuration). This makes dynamically created interfaces work in the same way as the default. Implementing this option requires non-standard networking APIs and it is only available under Linux. On other platforms it falls-back to **--bind-interfaces** mode.

−y, --localise-queries
Return answers to DNS queries from /etc/hosts and **--interface-name** and **--dynamic-host** which depend on the interface over which the query was received. If a name has more than one address associated with it, and at least one of those addresses is on the same subnet as the interface to which the query was sent, then return only the address(es) on that subnet. This allows for a server to have multiple addresses in /etc/hosts corresponding to each of its interfaces, and hosts will get the correct address based on which network they are attached to. Currently this facility is limited to IPv4.

−b, --bogus-priv
Bogus private reverse lookups. All reverse lookups for private IP ranges (ie 192.168.x.x, etc) which are not found in /etc/hosts or the DHCP leases file are answered with "no such domain" rather than being forwarded upstream. The set of prefixes affected is the list given in RFC6303, for IPv4 and IPv6.

−V, --alias=[<old-ip>]|[<start-ip>-<end-ip>],<new-ip>[,<mask>]

Modify IPv4 addresses returned from upstream nameservers; old-ip is replaced by new-ip. If the optional mask is given then any address which matches the masked old-ip will be re-written. So, for instance **--alias=1.2.3.0,6.7.8.0,255.255.255.0** will map 1.2.3.56 to 6.7.8.56 and 1.2.3.67 to 6.7.8.67. This is what Cisco PIX routers call "DNS doctoring". If the old IP is given as range, then only addresses in the range, rather than a whole subnet, are re-written. So **--alias=192.168.0.10-192.168.0.40,10.0.0.0,255.255.255.0** maps 192.168.0.10->192.168.0.40 to 10.0.0.10->10.0.0.40

−B, --bogus-nxdomain=<ipaddr>[/prefix]

Transform replies which contain the specified address or subnet into "No such domain" replies. IPv4 and IPv6 are supported. This is intended to counteract a devious move made by Verisign in September 2003 when they started returning the address of an advertising web page in response to queries for unregistered names, instead of the correct NXDOMAIN response. This option tells dnsmasq to fake the correct response when it sees this behaviour. As at Sept 2003 the IP address being returned by Verisign is 64.94.110.11

--ignore-address=<ipaddr>[/prefix]

Ignore replies to A or AAAA queries which include the specified address or subnet. No error is generated, dnsmasq simply continues to listen for another reply. This is useful to defeat blocking strategies which rely on quickly supplying a forged answer to a DNS request for certain domain, before the correct answer can arrive.

−f, --filterwin2k

Later versions of windows make periodic DNS requests which don't get sensible answers from the public DNS and can cause problems by triggering dial-on-demand links. This flag turns on an option to filter such requests. The requests blocked are for records of types SOA and SRV, and type ANY where the requested name has underscores, to catch LDAP requests.

−r, --resolv-file=<file>

Read the IP addresses of the upstream nameservers from <file>, instead of /etc/resolv.conf. For the format of this file see **resolv.conf**(5). The only lines relevant to dnsmasq are nameserver ones. Dnsmasq can be told to poll more than one resolv.conf file, the first file name specified overrides the default, subsequent ones add to the list. This is only allowed when polling; the file with the currently latest modification time is the one used.

−R, --no-resolv

Don't read /etc/resolv.conf. Get upstream servers only from the command line or the dnsmasq configuration file.

−1, --enable-dbus[=<service-name>]

Allow dnsmasq configuration to be updated via DBus method calls. The configuration which can be changed is upstream DNS servers (and corresponding domains) and cache clear. Requires that dnsmasq has been built with DBus support. If the service name is given, dnsmasq provides service at that name, rather than the default which is **uk.org.thekelleys.dnsmasq**

--enable-ubus[=<service-name>]

Enable dnsmasq UBus interface. It sends notifications via UBus on DHCPACK and DHCPRELEASE events. Furthermore it offers metrics and allows configuration of Linux connection track mark based filtering. When DNS query filtering based on Linux connection track marks is enabled UBus notifications are generated for each resolved or filtered DNS query. Requires that dnsmasq has been built with UBus support. If the service name is given, dnsmasq provides service at that namespace, rather than the default which is **dnsmasq**

−o, --strict-order

By default, dnsmasq will send queries to any of the upstream servers it knows about and tries to favour servers that are known to be up. Setting this flag forces dnsmasq to try each query with each server strictly in the order they appear in /etc/resolv.conf

--all-servers
By default, when dnsmasq has more than one upstream server available, it will send queries to just one server. Setting this flag forces dnsmasq to send all queries to all available servers. The reply from the server which answers first will be returned to the original requester.

--dns-loop-detect
Enable code to detect DNS forwarding loops; ie the situation where a query sent to one of the upstream server eventually returns as a new query to the dnsmasq instance. The process works by generating TXT queries of the form <hex>.test and sending them to each upstream server. The hex is a UID which encodes the instance of dnsmasq sending the query and the upstream server to which it was sent. If the query returns to the server which sent it, then the upstream server through which it was sent is disabled and this event is logged. Each time the set of upstream servers changes, the test is re-run on all of them, including ones which were previously disabled.

--stop-dns-rebind
Reject (and log) addresses from upstream nameservers which are in the private ranges. This blocks an attack where a browser behind a firewall is used to probe machines on the local network. For IPv6, the private range covers the IPv4-mapped addresses in private space plus all link-local (LL) and site-local (ULA) addresses.

--rebind-localhost-ok
Exempt 127.0.0.0/8 and ::1 from rebinding checks. This address range is returned by realtime black hole servers, so blocking it may disable these services.

--rebind-domain-ok=[<domain>]|[[/<domain>/[<domain>/]
Do not detect and block dns-rebind on queries to these domains. The argument may be either a single domain, or multiple domains surrounded by '/', like the **--server** syntax, eg. **--rebind-domain-ok=/domain1/domain2/domain3/**

−n, --no-poll
Don't poll /etc/resolv.conf for changes.

--clear-on-reload
Whenever /etc/resolv.conf is re-read or the upstream servers are set via DBus, clear the DNS cache. This is useful when new nameservers may have different data than that held in cache.

−D, --domain-needed
Tells dnsmasq to never forward A or AAAA queries for plain names, without dots or domain parts, to upstream nameservers. If the name is not known from /etc/hosts or DHCP then a "not found" answer is returned.

−S, --local, --server=[/[<domain>]/[domain/]][<ipaddr>[#<port>]][@<interface>][@<source-ip>[#<port>]]
Specify IP address of upstream servers directly. Setting this flag does not suppress reading of /etc/resolv.conf, use **--no-resolv** to do that. If one or more optional domains are given, that server is used only for those domains and they are queried only using the specified server. This is intended for private nameservers: if you have a nameserver on your network which deals with names of the form xxx.internal.thekelleys.org.uk at 192.168.1.1 then giving the flag**--ser ver=/internal.thekelleys.org.uk/192.168.1.1** will send all queries for internal machines to that nameserver, everything else will go to the servers in /etc/resolv.conf. DNSSEC validation is turned off for such private nameservers, UNLESS a **--trust-anchor** is specified for the domain in question. An empty domain specification, // has the special meaning of "unqualified names only" ie names without any dots in them. A non-standard port may be specified as part of the IP address using a # character. More than one **--server** flag is allowed, with repeated domain or ipaddr parts as required.

More specific domains take precedence over less specific domains, so: **--server=/google.com/1.2.3.4 --server=/www.google.com/2.3.4.5** will send queries for google.com and gmail.google.com to 1.2.3.4, but www.google.com will go to 2.3.4.5

Matching of domains is normally done on complete labels, so /google.com/ matches google.com and www.google.com but NOT supergoogle.com. This can be overridden with a * at the start of a pattern only: /*google.com/ will match google.com and www.google.com AND supergoogle.com. The non-wildcard form has priority, so if /google.com/ and /*google.com/ are both specified then google.com and www.google.com will match /google.com/ and /*google.com/ will only match supergoogle.com.

For historical reasons, the pattern /.google.com/ is equivalent to /google.com/ if you wish to match any subdomain of google.com but NOT google.com itself, use /*.google.com/

The special server address '#' means, "use the standard servers", so **--server=/google.com/1.2.3.4 --server=/www.google.com/#** will send queries for google.com and its subdomains to 1.2.3.4, except www.google.com (and its subdomains) which will be forwarded as usual.

Also permitted is a -S flag which gives a domain but no IP address; this tells dnsmasq that a domain is local and it may answer queries from /etc/hosts or DHCP but should never forward queries on that domain to any upstream servers. **--local** is a synonym for **--server** to make configuration files clearer in this case.

IPv6 addresses may include an %interface scope-id, eg fe80::202:a412:4512:7bbf%eth0.

The optional string after the @ character tells dnsmasq how to set the source of the queries to this nameserver. It can either be an ip-address, an interface name or both. The ip-address should belong to the machine on which dnsmasq is running, otherwise this server line will be logged and then ignored. If an interface name is given, then queries to the server will be forced via that interface; if an ip-address is given then the source address of the queries will be set to that address; and if both are given then a combination of ip-address and interface name will be used to steer requests to the server. The query-port flag is ignored for any servers which have a source address specified but the port may be specified directly as part of the source address. Forcing queries to an interface is not implemented on all platforms supported by dnsmasq.

--rev-server=<ip-address>/<prefix-len>[,<ipaddr>][#<port>][@<interface>][@<source-ip>[#<port>]]
This is functionally the same as **--server,** but provides some syntactic sugar to make specifying address-to-name queries easier. For example **--rev-server=1.2.3.0/24,192.168.0.1** is exactly equivalent to **--server=/3.2.1.in-addr.arpa/192.168.0.1**

−A, --address=/<domain>[/<domain>...]/[<ipaddr>]
Specify an IP address to return for any host in the given domains. Queries in the domains are never forwarded and always replied to with the specified IP address which may be IPv4 or IPv6. To give both IPv4 and IPv6 addresses for a domain, use repeated **--address** flags. To include multiple IP addresses for a single query, use **--addn-hosts=<path>** instead. Note that /etc/hosts and DHCP leases override this for individual names. A common use of this is to redirect the entire doubleclick.net domain to some friendly local web server to avoid banner ads. The domain specification works in the same way as for **--server**, with the additional facility that **/#/** matches any domain. Thus **--address=/#/1.2.3.4** will always return **1.2.3.4** for any query not answered from **/etc/hosts** or DHCP and not sent to an upstream nameserver by a more specific **--server** directive. As for **--server**, one or more domains with no address returns a no-such-domain answer, so **--address=/example.com/** is equivalent to **--server=/example.com/** and returns NXDOMAIN for example.com and all its subdomains. An address specified as '#' translates to the NULL address of 0.0.0.0 and its IPv6 equivalent of :: so **--address=/example.com/#** will return NULL addresses for example.com and its subdomains. This is partly syntactic sugar for **--address=/example.com/0.0.0.0** and **--address=/example.com/::** but is also more efficient than including both as separate configuration lines. Note that NULL addresses normally work in the same way as localhost, so beware that clients looking up these names are likely to end up talking to themselves.

--ipset=/<domain>[/<domain>...]/<ipset>[,<ipset>...]

Places the resolved IP addresses of queries for one or more domains in the specified Netfilter IP set. If multiple setnames are given, then the addresses are placed in each of them, subject to the limitations of an IP set (IPv4 addresses cannot be stored in an IPv6 IP set and vice versa). Domains and subdomains are matched in the same way as **--address**. These IP sets must already exist. See **ipset**(8) for more details.

--connmark-allowlist-enable[=<mask>]

Enables filtering of incoming DNS queries with associated Linux connection track marks according to individual allowlists configured via a series of **--connmark-allowlist** options. Disallowed queries are not forwarded; they are rejected with a REFUSED error code. DNS queries are only allowed if they do not have an associated Linux connection track mark, or if the queried domains match the configured DNS patterns for the associated Linux connection track mark. If no allowlist is configured for a Linux connection track mark, all DNS queries associated with that mark are rejected. If a mask is specified, Linux connection track marks are first bitwise ANDed with the given mask before being processed.

--connmark-allowlist=<connmark>[/<mask>][,<pattern>[/<pattern>...]]

Configures the DNS patterns that are allowed in DNS queries associated with the given Linux connection track mark. If a mask is specified, Linux connection track marks are first bitwise ANDed with the given mask before they are compared to the given connection track mark. Patterns follow the syntax of DNS names, but additionally allow the wildcard character "*" to be used up to twice per label to match 0 or more characters within that label. Note that the wildcard never matches a dot (e.g., "*.example.com" matches "api.example.com" but not "api.us.example.com"). Patterns must be fully qualified, i.e., consist of at least two labels. The final label must not be fully numeric, and must not be the "local" pseudo-TLD. A pattern must end with at least two literal (non-wildcard) labels. Instead of a pattern, "*" can be specified to disable allowlist filtering for a given Linux connection track mark entirely.

−m, --mx-host=<mx name>[[,<hostname>],<preference>]

Return an MX record named <mx name> pointing to the given hostname (if given), or the host specified in the **--mx-target** switch or, if that switch is not given, the host on which dnsmasq is running. The default is useful for directing mail from systems on a LAN to a central server. The preference value is optional, and defaults to 1 if not given. More than one MX record may be given for a host.

−t, --mx-target=<hostname>

Specify the default target for the MX record returned by dnsmasq. See **--mx-host**. If**--mx-tar get** is given, but not **--mx-host**, then dnsmasq returns a MX record containing the MX target for MX queries on the hostname of the machine on which dnsmasq is running.

−e, --selfmx

Return an MX record pointing to itself for each local machine. Local machines are those in /etc/hosts or with DHCP leases.

−L, --localmx

Return an MX record pointing to the host given by **--mx-target** (or the machine on which dnsmasq is running) for each local machine. Local machines are those in /etc/hosts or with DHCP leases.

−W, --srv-host=<_service>.<_prot>.[<domain>],[<target>[,<port>[,<priority>[,<weight>]]]]

Return a SRV DNS record. See RFC2782 for details. If not supplied, the domain defaults to that given by **--domain.** The default for the target domain is empty, and the default for port is one and the defaults for weight and priority are zero. Be careful if transposing data from BIND zone files: the port, weight and priority numbers are in a different order. More than one SRV record for a given service/domain is allowed, all that match are returned.

--host-record=<name>[,<name>....],[<IPv4-address>],[<IPv6-address>][,<TTL>]
Add A, AAAA and PTR records to the DNS. This adds one or more names to the DNS with associated IPv4 (A) and IPv6 (AAAA) records. A name may appear in more than one **--host-record** and therefore be assigned more than one address. Only the first address creates a PTR record linking the address to the name. This is the same rule as is used reading hosts-files. **--host-r ecord** options are considered to be read before host-files, so a name appearing there inhibits PTR-record creation if it appears in hosts-file also. Unlike hosts-files, names are not expanded, even when **--expand-hosts** is in effect. Short and long names may appear in the same **--host-record,** eg. **--host-record=laptop,laptop.thekelleys.org,192.168.0.1,1234::100**

If the time-to-live is given, it overrides the default, which is zero or the value of **--local-ttl**. The value is a positive integer and gives the time-to-live in seconds.

--dynamic-host=<name>,[IPv4-address],[IPv6-address],<interface>
Add A, AAAA and PTR records to the DNS in the same subnet as the specified interface. The address is derived from the network part of each address associated with the interface, and the host part from the specified address. For example **--dynamic-host=example.com,0.0.0.8,eth0** will, when eth0 has the address 192.168.78.x and netmask 255.255.255.0 give the name example.com an A record for 192.168.78.8. The same principle applies to IPv6 addresses. Note that if an interface has more than one address, more than one A or AAAA record will be created. The TTL of the records is always zero, and any changes to interface addresses will be immediately reflected in them.

−Y, --txt-record=<name>[[,<text>],<text>]
Return a TXT DNS record. The value of TXT record is a set of strings, so any number may be included, delimited by commas; use quotes to put commas into a string. Note that the maximum length of a single string is 255 characters, longer strings are split into 255 character chunks.

--ptr-record=<name>[,<target>]
Return a PTR DNS record.

--naptr-record=<name>,<order>,<preference>,<flags>,<service>,<regexp>[,<replacement>]
Return an NAPTR DNS record, as specified in RFC3403.

--caa-record=<name>,<flags>,<tag>,<value>
Return a CAA DNS record, as specified in RFC6844.

--cname=<cname>,[<cname>,]<target>[,<TTL>]
Return a CNAME record which indicates that <cname> is really <target>. There is a significant limitation on the target; it must be a DNS record which is known to dnsmasq and NOT a DNS record which comes from an upstream server. The cname must be unique, but it is permissible to have more than one cname pointing to the same target. Indeed it's possible to declare multiple cnames to a target in a single line, like so: **--cname=cname1,cname2,target**

If the time-to-live is given, it overrides the default, which is zero or the value of **--local-ttl**. The value is a positive integer and gives the time-to-live in seconds.

--dns-rr=<name>,<RR-number>,[<hex data>]
Return an arbitrary DNS Resource Record. The number is the type of the record (which is always in the C_IN class). The value of the record is given by the hex data, which may be of the form 01:23:45 or 01 23 45 or 012345 or any mixture of these.

--interface-name=<name>,<interface>[/4|/6]
Return DNS records associating the name with the address(es) of the given interface. This flag specifies an A or AAAA record for the given name in the same way as an /etc/hosts line, except that the address is not constant, but taken from the given interface. The interface may be followed by "/4" or "/6" to specify that only IPv4 or IPv6 addresses of the interface should be used. If the interface is down, not configured or non-existent, an empty record is returned. The matching PTR record is also created, mapping the interface address to the name. More than one name may be

associated with an interface address by repeating the flag; in that case the first instance is used for the reverse address-to-name mapping. Note that a name used in **--interface-name** may not appear in /etc/hosts.

--synth-domain=<domain>,<address range>[,<prefix>[*]]
Create artificial A/AAAA and PTR records for an address range. The records either seqential numbers or the address, with periods (or colons for IPv6) replaced with dashes.

An examples should make this clearer. First sequential numbers. **--synth-domain=thekelleys.org.uk,192.168.0.50,192.168.0.70,internal-*** results in the name internal-0.thekelleys.org.uk. returning 192.168.0.50, internal-1.thekelleys.org.uk returning 192.168.0.51 and so on. (note the *) The same principle applies to IPv6 addresses (where the numbers may be very large). Reverse lookups from address to name behave as expected.

Second, **--synth-domain=thekelleys.org.uk,192.168.0.0/24,internal- (no *)** will result in a query for internal-192-168-0-56.thekelleys.org.uk returning 192.168.0.56 and a reverse query vice versa. The same applies to IPv6, but IPv6 addresses may start with '::' but DNS labels may not start with '-' so in this case if no prefix is configured a zero is added in front of the label. ::1 becomes 0--1.

V4 mapped IPv6 addresses, which have a representation like ::ffff:1.2.3.4 are handled specially, and become like 0--ffff-1-2-3-4

The address range can be of the form <start address>,<end address> or <ip address>/<prefix-length> in both forms of the option. For IPv6 the start and end addresses must fall in the same /64 network, or prefix-length must be greater than or equal to 64 except that shorter prefix lengths than 64 are allowed only if non-sequential names are in use.

--dumpfile=<path/to/file>
Specify the location of a pcap-format file which dnsmasq uses to dump copies of network packets for debugging purposes. If the file exists when dnsmasq starts, it is not deleted; new packets are added to the end.

--dumpmask=<mask>
Specify which types of packets should be added to the dumpfile. The argument should be the OR of the bitmasks for each type of packet to be dumped: it can be specified in hex by preceding the number with 0x in the normal way. Each time a packet is written to the dumpfile, dnsmasq logs the packet sequence and the mask representing its type. The current types are: 0x0001 - DNS queries from clients 0x0002 DNS replies to clients 0x0004 - DNS queries to upstream 0x0008 - DNS replies from upstream 0x0010 - queries send upstream for DNSSEC validation 0x0020 - replies to queries for DNSSEC validation 0x0040 - replies to client queries which fail DNSSEC validation 0x0080 replies to queries for DNSSEC validation which fail validation.

--add-mac[=base64|text]
Add the MAC address of the requestor to DNS queries which are forwarded upstream. This may be used to DNS filtering by the upstream server. The MAC address can only be added if the requestor is on the same subnet as the dnsmasq server. Note that the mechanism used to achieve this (an EDNS0 option) is not yet standardised, so this should be considered experimental. Also note that exposing MAC addresses in this way may have security and privacy implications. The warning about caching given for **--add-subnet** applies to **--add-mac** too. An alternative encoding of the MAC, as base64, is enabled by adding the "base64" parameter and a human-readable encoding of hex-and-colons is enabled by added the "text" parameter.

--add-cpe-id=<string>
Add an arbitrary identifying string to DNS queries which are forwarded upstream.

--add-subnet[[=[<IPv4 address>/]<IPv4 prefix length>][,[<IPv6 address>/]<IPv6 prefix length>]]
Add a subnet address to the DNS queries which are forwarded upstream. If an address is specified in the flag, it will be used, otherwise, the address of the requestor will be used. The amount of the

address forwarded depends on the prefix length parameter: 32 (128 for IPv6) forwards the whole address, zero forwards none of it but still marks the request so that no upstream nameserver will add client address information either. The default is zero for both IPv4 and IPv6. Note that upstream nameservers may be configured to return different results based on this information, but the dnsmasq cache does not take account. Caching is therefore disabled for such replies, unless the subnet address being added is constant.

For example, **--add-subnet=24,96** will add the /24 and /96 subnets of the requestor for IPv4 and IPv6 requestors, respectively. **--add-subnet=1.2.3.4/24** will add 1.2.3.0/24 for IPv4 requestors and ::/0 for IPv6 requestors. **--add-subnet=1.2.3.4/24,1.2.3.4/24** will add 1.2.3.0/24 for both IPv4 and IPv6 requestors.

--umbrella[=deviceid:<deviceid>[,orgid:<orgid>]]
Embeds the requestor's IP address in DNS queries forwarded upstream. If device id or organization id are specified, the information is included in the forwarded queries and may be able to be used in filtering policies and reporting. The order of the deviceid and orgid attributes is irrelevant, but must be separated by a comma.

−c, --cache-size=<cachesize>
Set the size of dnsmasq's cache. The default is 150 names. Setting the cache size to zero disables caching. Note: huge cache size impacts performance.

−N, --no-negcache
Disable negative caching. Negative caching allows dnsmasq to remember "no such domain" answers from upstream nameservers and answer identical queries without forwarding them again.

−0, --dns-forward-max=<queries>
Set the maximum number of concurrent DNS queries. The default value is 150, which should be fine for most setups. The only known situation where this needs to be increased is when using web-server log file resolvers, which can generate large numbers of concurrent queries. This parameter actually controls the number of concurrent queries per server group, where a server group is the set of server(s) associated with a single domain. So if a domain has it's own server via --server=/example.com/1.2.3.4 and 1.2.3.4 is not responding, but queries for *.example.com cannot go elsewhere, then other queries will not be affected. On configurations with many such server groups and tight resources, this value may need to be reduced.

--dnssec
Validate DNS replies and cache DNSSEC data. When forwarding DNS queries, dnsmasq requests the DNSSEC records needed to validate the replies. The replies are validated and the result returned as the Authenticated Data bit in the DNS packet. In addition the DNSSEC records are stored in the cache, making validation by clients more efficient. Note that validation by clients is the most secure DNSSEC mode, but for clients unable to do validation, use of the AD bit set by dnsmasq is useful, provided that the network between the dnsmasq server and the client is trusted. Dnsmasq must be compiled with HAVE_DNSSEC enabled, and DNSSEC trust anchors provided, see **--trust-anchor.** Because the DNSSEC validation process uses the cache, it is not permitted to reduce the cache size below the default when DNSSEC is enabled. The nameservers upstream of dnsmasq must be DNSSEC-capable, ie capable of returning DNSSEC records with data. If they are not, then dnsmasq will not be able to determine the trusted status of answers and this means that DNS service will be entirely broken.

--trust-anchor=[<class>],<domain>,<key-tag>,<algorithm>,<digest-type>,<digest>
Provide DS records to act a trust anchors for DNSSEC validation. Typically these will be the DS record(s) for Key Signing key(s) (KSK) of the root zone, but trust anchors for limited domains are also possible. The current root-zone trust anchors may be downloaded from https://data.iana.org/root-anchors/root-anchors.xml

--dnssec-check-unsigned[=no]
As a default, dnsmasq checks that unsigned DNS replies are legitimate: this entails possible extra queries even for the majority of DNS zones which are not, at the moment, signed. If **--dnssec-**

check-unsigned=no appears in the configuration, then such replies they are assumed to be valid and passed on (without the "authentic data" bit set, of course). This does not protect against an attacker forging unsigned replies for signed DNS zones, but it is fast.

Versions of dnsmasq prior to 2.80 defaulted to not checking unsigned replies, and used **--dnssec-check-unsigned** to switch this on. Such configurations will continue to work as before, but those which used the default of no checking will need to be altered to explicitly select no checking. The new default is because switching off checking for unsigned replies is inherently dangerous. Not only does it open the possiblity of forged replies, but it allows everything to appear to be working even when the upstream namesevers do not support DNSSEC, and in this case no DNSSEC validation at all is occurring.

--dnssec-no-timecheck

DNSSEC signatures are only valid for specified time windows, and should be rejected outside those windows. This generates an interesting chicken-and-egg problem for machines which don't have a hardware real time clock. For these machines to determine the correct time typically requires use of NTP and therefore DNS, but validating DNS requires that the correct time is already known. Setting this flag removes the time-window checks (but not other DNSSEC validation.) only until the dnsmasq process receives SIGINT. The intention is that dnsmasq should be started with this flag when the platform determines that reliable time is not currently available. As soon as reliable time is established, a SIGINT should be sent to dnsmasq, which enables time checking, and purges the cache of DNS records which have not been thoroughly checked.

Earlier versions of dnsmasq overloaded SIGHUP (which re-reads much configuration) to also enable time validation.

If dnsmasq is run in debug mode (**--no-daemon** flag) then SIGINT retains its usual meaning of terminating the dnsmasq process.

--dnssec-timestamp=<path>

Enables an alternative way of checking the validity of the system time for DNSSEC (see **--dnssec-no-timecheck**). In this case, the system time is considered to be valid once it becomes later than the timestamp on the specified file. The file is created and its timestamp set automatically by dnsmasq. The file must be stored on a persistent filesystem, so that it and its mtime are carried over system restarts. The timestamp file is created after dnsmasq has dropped root, so it must be in a location writable by the unprivileged user that dnsmasq runs as.

--proxy-dnssec

Copy the DNSSEC Authenticated Data bit from upstream servers to downstream clients. This is an alternative to having dnsmasq validate DNSSEC, but it depends on the security of the network between dnsmasq and the upstream servers, and the trustworthiness of the upstream servers. Note that caching the Authenticated Data bit correctly in all cases is not technically possible. If the AD bit is to be relied upon when using this option, then the cache should be disabled using --cache-size=0. In most cases, enabling DNSSEC validation within dnsmasq is a better option. See --dnssec for details.

--dnssec-debug

Set debugging mode for the DNSSEC validation, set the Checking Disabled bit on upstream queries, and don't convert replies which do not validate to responses with a return code of SERVFAIL. Note that setting this may affect DNS behaviour in bad ways, it is not an extra-logging flag and should not be set in production.

--auth-zone=<domain>[,<subnet>[/<prefix length>][,<subnet>[/<prefix length>].....][,exclude:<subnet>[/<prefix length>]].....]

Define a DNS zone for which dnsmasq acts as authoritative server. Locally defined DNS records which are in the domain will be served. If subnet(s) are given, A and AAAA records must be in one of the specified subnets.

As alternative to directly specifying the subnets, it's possible to give the name of an interface, in which case the subnets implied by that interface's configured addresses and netmask/prefix-length are used; this is useful when using constructed DHCP ranges as the actual address is dynamic and not known when configuring dnsmasq. The interface addresses may be confined to only IPv6 addresses using <interface>/6 or to only IPv4 using <interface>/4. This is useful when an interface has dynamically determined global IPv6 addresses which should appear in the zone, but RFC1918 IPv4 addresses which should not. Interface-name and address-literal subnet specifications may be used freely in the same **--auth-zone** declaration.

It's possible to exclude certain IP addresses from responses. It can be used, to make sure that answers contain only global routeable IP addresses (by excluding loopback, RFC1918 and ULA addresses).

The subnet(s) are also used to define in-addr.arpa and ip6.arpa domains which are served for reverse-DNS queries. If not specified, the prefix length defaults to 24 for IPv4 and 64 for IPv6. For IPv4 subnets, the prefix length should be have the value 8, 16 or 24 unless you are familiar with RFC 2317 and have arranged the in-addr.arpa delegation accordingly. Note that if no subnets are specified, then no reverse queries are answered.

--auth-soa=<serial>[,<hostmaster>[,<refresh>[,<retry>[,<expiry>]]]]

Specify fields in the SOA record associated with authoritative zones. Note that this is optional, all the values are set to sane defaults.

--auth-sec-servers=<domain>[,<domain>[,<domain>...]]

Specify any secondary servers for a zone for which dnsmasq is authoritative. These servers must be configured to get zone data from dnsmasq by zone transfer, and answer queries for the same authoritative zones as dnsmasq.

--auth-peer=<ip-address>[,<ip-address>[,<ip-address>...]]

Specify the addresses of secondary servers which are allowed to initiate zone transfer (AXFR) requests for zones for which dnsmasq is authoritative. If this option is not given but --auth-sec-servers is, then AXFR requests will be accepted from any secondary. Specifying **--auth-peer** without **--auth-sec-servers** enables zone transfer but does not advertise the secondary in NS records returned by dnsmasq.

--conntrack

Read the Linux connection track mark associated with incoming DNS queries and set the same mark value on upstream traffic used to answer those queries. This allows traffic generated by dnsmasq to be associated with the queries which cause it, useful for bandwidth accounting and firewalling. Dnsmasq must have conntrack support compiled in and the kernel must have conntrack support included and configured. This option cannot be combined with **--query-port.**

−F, --dhcp-range=[tag:<tag>[,tag:<tag>],][set:<tag>,]<start-addr>[,<end-addr>|<mode>[,<netmask>[,<broadcast>]]][,<lease time>]

−F, --dhcp-range=[tag:<tag>[,tag:<tag>],][set:<tag>,]<start-IPv6addr>[,<end-IPv6addr>|constructor:<interface>][,<mode>][,<prefix-len>][,<lease time>]

Enable the DHCP server. Addresses will be given out from the range <start-addr> to <end-addr> and from statically defined addresses given in **--dhcp-host** options. If the lease time is given, then leases will be given for that length of time. The lease time is in seconds, or minutes (eg 45m) or hours (eg 1h) or days (2d) or weeks (1w) or "infinite". If not given, the default lease time is one hour for IPv4 and one day for IPv6. The minimum lease time is two minutes. For IPv6 ranges, the lease time maybe "deprecated"; this sets the preferred lifetime sent in a DHCP lease or router advertisement to zero, which causes clients to use other addresses, if available, for new connections as a prelude to renumbering.

This option may be repeated, with different addresses, to enable DHCP service to more than one

network. For directly connected networks (ie, networks on which the machine running dnsmasq has an interface) the netmask is optional: dnsmasq will determine it from the interface configuration. For networks which receive DHCP service via a relay agent, dnsmasq cannot determine the netmask itself, so it should be specified, otherwise dnsmasq will have to guess, based on the class (A, B or C) of the network address. The broadcast address is always optional. It is always allowed to have more than one **--dhcp-range** in a single subnet.

For IPv6, the parameters are slightly different: instead of netmask and broadcast address, there is an optional prefix length which must be equal to or larger then the prefix length on the local interface. If not given, this defaults to 64. Unlike the IPv4 case, the prefix length is not automatically derived from the interface configuration. The minimum size of the prefix length is 64.

IPv6 (only) supports another type of range. In this, the start address and optional end address contain only the network part (ie ::1) and they are followed by **constructor:<interface>.** This forms a template which describes how to create ranges, based on the addresses assigned to the interface. For instance

--dhcp-range=::1,::400,constructor:eth0

will look for addresses on eth0 and then create a range from <network>::1 to <network>::400. If the interface is assigned more than one network, then the corresponding ranges will be automatically created, and then deprecated and finally removed again as the address is deprecated and then deleted. The interface name may have a final "*" wildcard. Note that just any address on eth0 will not do: it must not be an autoconfigured or privacy address, or be deprecated.

If a **--dhcp-range** is only being used for stateless DHCP and/or SLAAC, then the address can be simply ::

--dhcp-range=::,constructor:eth0

The optional **set:<tag>** sets an alphanumeric label which marks this network so that DHCP options may be specified on a per-network basis. When it is prefixed with 'tag:' instead, then its meaning changes from setting a tag to matching it. Only one tag may be set, but more than one tag may be matched.

The optional <mode> keyword may be **static** which tells dnsmasq to enable DHCP for the network specified, but not to dynamically allocate IP addresses: only hosts which have static addresses given via **--dhcp-host** or from /etc/ethers will be served. A static-only subnet with address all zeros may be used as a "catch-all" address to enable replies to all Information-request packets on a subnet which is provided with stateless DHCPv6, ie **--dhcp-range=::,static**

For IPv4, the <mode> may be **proxy** in which case dnsmasq will provide proxy-DHCP on the specified subnet. (See **--pxe-prompt** and **--pxe-service** for details.)

For IPv6, the mode may be some combination of **ra-only, slaac, ra-names, ra-stateless, ra-advrouter, off-link.**

ra-only tells dnsmasq to offer Router Advertisement only on this subnet, and not DHCP.

slaac tells dnsmasq to offer Router Advertisement on this subnet and to set the A bit in the router advertisement, so that the client will use SLAAC addresses. When used with a DHCP range or static DHCP address this results in the client having both a DHCP-assigned and a SLAAC address.

ra-stateless sends router advertisements with the O and A bits set, and provides a stateless DHCP service. The client will use a SLAAC address, and use DHCP for other configuration information.

ra-names enables a mode which gives DNS names to dual-stack hosts which do SLAAC for IPv6. Dnsmasq uses the host's IPv4 lease to derive the name, network segment and MAC address and assumes that the host will also have an IPv6 address calculated using the SLAAC algorithm, on the same network segment. The address is pinged, and if a reply is received, an AAAA record is added to the DNS for this IPv6 address. Note that this is only happens for directly-connected networks, (not one doing DHCP via a relay) and it will not work if a host is using privacy extensions. **ra-names** can be combined with **ra-stateless** and **slaac.**

ra-advrouter enables a mode where router address(es) rather than prefix(es) are included in the advertisements. This is described in RFC-3775 section 7.2 and is used in mobile IPv6. In this mode the interval option is also included, as described in RFC-3775 section 7.3.

off-link tells dnsmasq to advertise the prefix without the on-link (aka L) bit set.

−G, --dhcp-host=[<hwaddr>][,id:<client_id>|*][,set:<tag>][tag:<tag>][,<ipaddr>][,<hostname>][,<lease_time>][,ignore]

Specify per host parameters for the DHCP server. This allows a machine with a particular hardware address to be always allocated the same hostname, IP address and lease time. A hostname specified like this overrides any supplied by the DHCP client on the machine. It is also allowable to omit the hardware address and include the hostname, in which case the IP address and lease times will apply to any machine claiming that name. For example **--dhcp-host=00:20:e0:3b:13:af,wap,infinite** tells dnsmasq to give the machine with hardware address 00:20:e0:3b:13:af the name wap, and an infinite DHCP lease. **--dhcp-host=lap,192.168.0.199** tells dnsmasq to always allocate the machine lap the IP address 192.168.0.199.

Addresses allocated like this are not constrained to be in the range given by the **--dhcp-range** option, but they must be in the same subnet as some valid dhcp-range. For subnets which don't need a pool of dynamically allocated addresses, use the "static" keyword in the **--dhcp-range** declaration.

It is allowed to use client identifiers (called client DUID in IPv6-land) rather than hardware addresses to identify hosts by prefixing with 'id:'. Thus: **--dhcp-host=id:01:02:03:04,.....** refers to the host with client identifier 01:02:03:04. It is also allowed to specify the client ID as text, like this: **--dhcp-host=id:clientidastext,.....**

A single **--dhcp-host** may contain an IPv4 address or one or more IPv6 addresses, or both. IPv6 addresses must be bracketed by square brackets thus: **--dhcp-host=laptop,[1234::56]** IPv6 addresses may contain only the host-identifier part: **--dhcp-host=laptop,[::56]** in which case they act as wildcards in constructed DHCP ranges, with the appropriate network part inserted. For IPv6, an address may include a prefix length: **--dhcp-host=laptop,[1234:50/126]** which (in this case) specifies four addresses, 1234::50 to 1234::53. This (an the ability to specify multiple addresses) is useful when a host presents either a consistent name or hardware-ID, but varying DUIDs, since it allows dnsmasq to honour the static address allocation but assign a different adddress for each DUID. This typically occurs when chain netbooting, as each stage of the chain gets in turn allocates an address.

Note that in IPv6 DHCP, the hardware address may not be available, though it normally is for direct-connected clients, or clients using DHCP relays which support RFC 6939.

For DHCPv4, the special option id:* means "ignore any client-id and use MAC addresses only."

This is useful when a client presents a client-id sometimes but not others.

If a name appears in /etc/hosts, the associated address can be allocated to a DHCP lease, but only if a **--dhcp-host** option specifying the name also exists. Only one hostname can be given in a **--dhcp-host** option, but aliases are possible by using CNAMEs. (See **--cname**).

More than one **--dhcp-host** can be associated (by name, hardware address or UID) with a host. Which one is used (and therefore which address is allocated by DHCP and appears in the DNS) depends on the subnet on which the host last obtained a DHCP lease: the **--dhcp-host** with an address within the subnet is used. If more than one address is within the subnet, the result is undefined. A corollary to this is that the name associated with a host using **--dhcp-host** does not appear in the DNS until the host obtains a DHCP lease.

The special keyword "ignore" tells dnsmasq to never offer a DHCP lease to a machine. The machine can be specified by hardware address, client ID or hostname, for instance **--dhcp-host=00:20:e0:3b:13:af,ignore** This is useful when there is another DHCP server on the network which should be used by some machines.

The set:<tag> construct sets the tag whenever this **--dhcp-host** directive is in use. This can be used to selectively send DHCP options just for this host. More than one tag can be set in a **--dhcp-host** directive (but not in other places where "set:<tag>" is allowed). When a host matches any **--dhcp-host** directive (or one implied by /etc/ethers) then the special tag "known" is set. This allows dnsmasq to be configured to ignore requests from unknown machines using **--dhcp-ignore=tag:!known** If the host matches only a **--dhcp-host** directive which cannot be used because it specifies an address on different subnet, the tag "known-othernet" is set.

The tag:<tag> construct filters which dhcp-host directives are used. Tagged directives are used in preference to untagged ones.

Ethernet addresses (but not client-ids) may have wildcard bytes, so for example **--dhcp-host=00:20:e0:3b:13:*,ignore** will cause dnsmasq to ignore a range of hardware addresses. Note that the "*" will need to be escaped or quoted on a command line, but not in the configuration file.

Hardware addresses normally match any network (ARP) type, but it is possible to restrict them to a single ARP type by preceding them with the ARP-type (in HEX) and "-". so **--dhcp-host=06-00:20:e0:3b:13:af,1.2.3.4** will only match a Token-Ring hardware address, since the ARP-address type for token ring is 6.

As a special case, in DHCPv4, it is possible to include more than one hardware address. eg: **--dhcp-host=11:22:33:44:55:66,12:34:56:78:90:12,192.168.0.2** This allows an IP address to be associated with multiple hardware addresses, and gives dnsmasq permission to abandon a DHCP lease to one of the hardware addresses when another one asks for a lease. Beware that this is a dangerous thing to do, it will only work reliably if only one of the hardware addresses is active at any time and there is no way for dnsmasq to enforce this. It is, for instance, useful to allocate a stable IP address to a laptop which has both wired and wireless interfaces.

--dhcp-hostsfile=<path>

Read DHCP host information from the specified file. If a directory is given, then read all the files contained in that directory in alphabetical order. The file contains information about one host per line. The format of a line is the same as text to the right of '=' in **--dhcp-host**. The advantage of storing DHCP host information in this file is that it can be changed without re-starting dnsmasq: the file will be re-read when dnsmasq receives SIGHUP.

--dhcp-optsfile=<path>

Read DHCP option information from the specified file. If a directory is given, then read all the files contained in that directory in alphabetical order. The advantage of using this option is the same as for **--dhcp-hostsfile**: the **--dhcp-optsfile** will be re-read when dnsmasq receives SIGHUP. Note that it is possible to encode the information in a **--dhcp-boot** flag as DHCP options, using the options names bootfile-name, server-ip-address and tftp-server. This allows these to be included in a **--dhcp-optsfile**.

--dhcp-hostsdir=<path>

This is equivalent to **--dhcp-hostsfile**, except for the following. The path MUST be a directory, and not an individual file. Changed or new files within the directory are read automatically, without the need to send SIGHUP. If a file is deleted or changed after it has been read by dnsmasq, then the host record it contained will remain until dnsmasq receives a SIGHUP, or is restarted; ie host records are only added dynamically. The order in which the files in a directory are read is not defined.

--dhcp-optsdir=<path>

This is equivalent to **--dhcp-optsfile**, with the differences noted for **--dhcp-hostsdir**.

−Z, --read-ethers

Read /etc/ethers for information about hosts for the DHCP server. The format of /etc/ethers is a hardware address, followed by either a hostname or dotted-quad IP address. When read by dnsmasq these lines have exactly the same effect as **--dhcp-host** options containing the same information. /etc/ethers is re-read when dnsmasq receives SIGHUP. IPv6 addresses are NOT read from /etc/ethers.

−O, --dhcp-option=[tag:<tag>,[tag:<tag>,]][encap:<opt>,][vi-encap:<enterprise>,][vendor:[<vendor-class>],][<opt>|option:<opt-name>|option6:<opt>|option6:<opt-name>],[<value>[,<value>]]

Specify different or extra options to DHCP clients. By default, dnsmasq sends some standard options to DHCP clients, the netmask and broadcast address are set to the same as the host running dnsmasq, and the DNS server and default route are set to the address of the machine running dnsmasq. (Equivalent rules apply for IPv6.) If the domain name option has been set, that is sent. This configuration allows these defaults to be overridden, or other options specified. The option, to be sent may be given as a decimal number or as "option:<option-name>" The option numbers are specified in RFC2132 and subsequent RFCs. The set of option-names known by dnsmasq can be discovered by running "dnsmasq --help dhcp". For example, to set the default route option to 192.168.4.4, do **--dhcp-option=3,192.168.4.4** or **--dhcp-option = option:router, 192.168.4.4** and to set the time-server address to 192.168.0.4, do **--dhcp-option = 42,192.168.0.4** or **--dhcp-option = option:ntp-server, 192.168.0.4** The special address 0.0.0.0 is taken to mean "the address of the machine running dnsmasq".

Data types allowed are comma separated dotted-quad IPv4 addresses, []-wrapped IPv6 addresses, a decimal number, colon-separated hex digits and a text string. If the optional tags are given then this option is only sent when all the tags are matched.

Special processing is done on a text argument for option 119, to conform with RFC 3397. Text or dotted-quad IP addresses as arguments to option 120 are handled as per RFC 3361. Dotted-quad IP addresses which are followed by a slash and then a netmask size are encoded as described in RFC 3442.

IPv6 options are specified using the **option6:** keyword, followed by the option number or option name. The IPv6 option name space is disjoint from the IPv4 option name space. IPv6 addresses in options must be bracketed with square brackets, eg. **--dhcp-option=option6:ntp-server,[1234::56]** For IPv6, [::] means "the global address of the machine running dnsmasq", whilst [fd00::] is replaced with the ULA, if it exists, and [fe80::] with the link-local address.

Be careful: no checking is done that the correct type of data for the option number is sent, it is

quite possible to persuade dnsmasq to generate illegal DHCP packets with injudicious use of this flag. When the value is a decimal number, dnsmasq must determine how large the data item is. It does this by examining the option number and/or the value, but can be overridden by appending a single letter flag as follows: b = one byte, s = two bytes, i = four bytes. This is mainly useful with encapsulated vendor class options (see below) where dnsmasq cannot determine data size from the option number. Option data which consists solely of periods and digits will be interpreted by dnsmasq as an IP address, and inserted into an option as such. To force a literal string, use quotes. For instance when using option 66 to send a literal IP address as TFTP server name, it is necessary to do **--dhcp-option=66,"1.2.3.4"**

Encapsulated Vendor-class options may also be specified (IPv4 only) using **--dhcp-option**: for instance **--dhcp-option=vendor:PXEClient,1,0.0.0.0** sends the encapsulated vendor class-specific option "mftp-address=0.0.0.0" to any client whose vendor-class matches "PXEClient". The vendor-class matching is substring based (see **--dhcp-vendorclass** for details). If a vendor-class option (number 60) is sent by dnsmasq, then that is used for selecting encapsulated options in preference to any sent by the client. It is possible to omit the vendorclass completely; **--dhcp-option=vendor:,1,0.0.0.0** in which case the encapsulated option is always sent.

Options may be encapsulated (IPv4 only) within other options: for instance **--dhcp-option=encap:175, 190, iscsi-client0** will send option 175, within which is the option 190. If multiple options are given which are encapsulated with the same option number then they will be correctly combined into one encapsulated option. encap: and vendor: are may not both be set in the same **--dhcp-option**.

The final variant on encapsulated options is "Vendor-Identifying Vendor Options" as specified by RFC3925. These are denoted like this: **--dhcp-option=vi-encap:2, 10, text** The number in the vi-encap: section is the IANA enterprise number used to identify this option. This form of encapsulation is supported in IPv6.

The address 0.0.0.0 is not treated specially in encapsulated options.

--dhcp-option-force=[tag:<tag>,[tag:<tag>,]][encap:<opt>,][vi-encap:<enterprise>,][vendor:[<vendor-class>],]<opt>,[<value>[,<value>]]

This works in exactly the same way as **--dhcp-option** except that the option will always be sent, even if the client does not ask for it in the parameter request list. This is sometimes needed, for example when sending options to PXELinux.

--dhcp-no-override

(IPv4 only) Disable re-use of the DHCP servername and filename fields as extra option space. If it can, dnsmasq moves the boot server and filename information (from **--dhcp-boot**) out of their dedicated fields into DHCP options. This make extra space available in the DHCP packet for options but can, rarely, confuse old or broken clients. This flag forces "simple and safe" behaviour to avoid problems in such a case.

--dhcp-relay=<local address>,<server address>[,<interface]

Configure dnsmasq to do DHCP relay. The local address is an address allocated to an interface on the host running dnsmasq. All DHCP requests arriving on that interface will we relayed to a remote DHCP server at the server address. It is possible to relay from a single local address to multiple remote servers by using multiple **--dhcp-relay** configs with the same local address and different server addresses. A server address must be an IP literal address, not a domain name. In the case of DHCPv6, the server address may be the ALL_SERVERS multicast address, ff05::1:3. In this case the interface must be given, not be wildcard, and is used to direct the multicast to the correct interface to reach the DHCP server.

Access control for DHCP clients has the same rules as for the DHCP server, see **--interface**, **--except-interface**, etc. The optional interface name in the **--dhcp-relay** config has a different

function: it controls on which interface DHCP replies from the server will be accepted. This is intended for configurations which have three interfaces: one being relayed from, a second connecting the DHCP server, and a third untrusted network, typically the wider internet. It avoids the possibility of spoof replies arriving via this third interface.

It is allowed to have dnsmasq act as a DHCP server on one set of interfaces and relay from a disjoint set of interfaces. Note that whilst it is quite possible to write configurations which appear to act as a server and a relay on the same interface, this is not supported: the relay function will take precedence.

Both DHCPv4 and DHCPv6 relay is supported. It's not possible to relay DHCPv4 to a DHCPv6 server or vice-versa.

−U, --dhcp-vendorclass=set:<tag>,[enterprise:<IANA-enterprise number>,]<vendor-class>
Map from a vendor-class string to a tag. Most DHCP clients provide a "vendor class" which represents, in some sense, the type of host. This option maps vendor classes to tags, so that DHCP options may be selectively delivered to different classes of hosts. For example **--dhcp-vendorclass=set:printers,Hewlett-Packard JetDirect** will allow options to be set only for HP printers like so: **--dhcp-option=tag:printers,3,192.168.4.4** The vendor-class string is substring matched against the vendor-class supplied by the client, to allow fuzzy matching. The set: prefix is optional but allowed for consistency.

Note that in IPv6 only, vendorclasses are namespaced with an IANA-allocated enterprise number. This is given with enterprise: keyword and specifies that only vendorclasses matching the specified number should be searched.

−j, --dhcp-userclass=set:<tag>,<user-class>
Map from a user-class string to a tag (with substring matching, like vendor classes). Most DHCP clients provide a "user class" which is configurable. This option maps user classes to tags, so that DHCP options may be selectively delivered to different classes of hosts. It is possible, for instance to use this to set a different printer server for hosts in the class "accounts" than for hosts in the class "engineering".

−4, --dhcp-mac=set:<tag>,<MAC address>
Map from a MAC address to a tag. The MAC address may include wildcards. For example **--dhcp-mac=set:3com,01:34:23:*:*:*** will set the tag "3com" for any host whose MAC address matches the pattern.

--dhcp-circuitid=set:<tag>,<circuit-id>, --dhcp-remoteid=set:<tag>,<remote-id>
Map from RFC3046 relay agent options to tags. This data may be provided by DHCP relay agents. The circuit-id or remote-id is normally given as colon-separated hex, but is also allowed to be a simple string. If an exact match is achieved between the circuit or agent ID and one provided by a relay agent, the tag is set.

--dhcp-remoteid (but not **--dhcp-circuitid**) is supported in IPv6.

--dhcp-subscrid=set:<tag>,<subscriber-id>
(IPv4 and IPv6) Map from RFC3993 subscriber-id relay agent options to tags.

--dhcp-proxy[=<ip addr>].....
(IPv4 only) A normal DHCP relay agent is only used to forward the initial parts of a DHCP interaction to the DHCP server. Once a client is configured, it communicates directly with the server. This is undesirable if the relay agent is adding extra information to the DHCP packets, such as that used by **--dhcp-circuitid** and **--dhcp-remoteid.** A full relay implementation can use the RFC 5107 serverid-override option to force the DHCP server to use the relay as a full proxy, with all packets passing through it. This flag provides an alternative method of doing the same thing, for relays which don't support RFC 5107. Given alone, it manipulates the server-id for all interactions via relays. If a list of IP addresses is given, only interactions via relays at those addresses are

affected.

--dhcp-match=set:<tag>,<option number>|option:<option name>|vi-encap:<enterprise>[,<value>]
Without a value, set the tag if the client sends a DHCP option of the given number or name. When a value is given, set the tag only if the option is sent and matches the value. The value may be of the form "01:ff:*:02" in which case the value must match (apart from wildcards) but the option sent may have unmatched data past the end of the value. The value may also be of the same form as in **--dhcp-option** in which case the option sent is treated as an array, and one element must match, so **--dhcp-match=set:efi-ia32,option:client-arch,6** will set the tag "efi-ia32" if the the number 6 appears in the list of architectures sent by the client in option 93. (See RFC 4578 for details.) If the value is a string, substring matching is used.

The special form with vi-encap:<enterprise number> matches against vendor-identifying vendor classes for the specified enterprise. Please see RFC 3925 for more details of these rare and interesting beasts.

--dhcp-name-match=set:<tag>,<name>[*]
Set the tag if the given name is supplied by a DHCP client. There may be a single trailing wildcard *, which has the usual meaning. Combined with dhcp-ignore or dhcp-ignore-names this gives the ability to ignore certain clients by name, or disallow certain hostnames from being claimed by a client.

--tag-if=set:<tag>[,set:<tag>[,tag:<tag>[,tag:<tag>]]]
Perform boolean operations on tags. Any tag appearing as set:<tag> is set if all the tags which appear as tag:<tag> are set, (or unset when tag:!<tag> is used) If no tag:<tag> appears set:<tag> tags are set unconditionally. Any number of set: and tag: forms may appear, in any order. **--tag-if** lines are executed in order, so if the tag in tag:<tag> is a tag set by another **--tag-if,** the line which sets the tag must precede the one which tests it.

As an extension, the tag:<tag> clauses support limited wildcard matching, similar to the matching in the **--interface** directive. This allows, for example, using **--tag-if=set:ppp,tag:ppp*** to set the tag 'ppp' for all requests received on any matching interface (ppp0, ppp1, etc). This can be used in conjunction with the tag:!<tag> format meaning that no tag matching the wildcard may be set.

−J, --dhcp-ignore=tag:<tag>[,tag:<tag>]
When all the given tags appear in the tag set ignore the host and do not allocate it a DHCP lease.

--dhcp-ignore-names[=tag:<tag>[,tag:<tag>]]
When all the given tags appear in the tag set, ignore any hostname provided by the host. Note that, unlike **--dhcp-ignore**, it is permissible to supply no tags, in which case DHCP-client supplied hostnames are always ignored, and DHCP hosts are added to the DNS using only **--dhcp-host** configuration in dnsmasq and the contents of /etc/hosts and /etc/ethers.

--dhcp-generate-names=tag:<tag>[,tag:<tag>]
(IPv4 only) Generate a name for DHCP clients which do not otherwise have one, using the MAC address expressed in hex, separated by dashes. Note that if a host provides a name, it will be used by preference to this, unless **--dhcp-ignore-names** is set.

--dhcp-broadcast[=tag:<tag>[,tag:<tag>]]
(IPv4 only) When all the given tags appear in the tag set, always use broadcast to communicate with the host when it is unconfigured. It is permissible to supply no tags, in which case this is unconditional. Most DHCP clients which need broadcast replies set a flag in their requests so that this happens automatically, some old BOOTP clients do not.

−M, --dhcp-boot=[tag:<tag>,]<filename>,[<servername>[,<server address>|<tftp_servername>]]
(IPv4 only) Set BOOTP options to be returned by the DHCP server. Server name and address are optional: if not provided, the name is left empty, and the address set to the address of the machine running dnsmasq. If dnsmasq is providing a TFTP service (see **--enable-tftp**) then only the filename is required here to enable network booting. If the optional tag(s) are given, they must match

for this configuration to be sent. Instead of an IP address, the TFTP server address can be given as a domain name which is looked up in /etc/hosts. This name can be associated in /etc/hosts with multiple IP addresses, which are used round-robin. This facility can be used to load balance the tftp load among a set of servers.

--dhcp-sequential-ip

Dnsmasq is designed to choose IP addresses for DHCP clients using a hash of the client's MAC address. This normally allows a client's address to remain stable long-term, even if the client sometimes allows its DHCP lease to expire. In this default mode IP addresses are distributed pseudo-randomly over the entire available address range. There are sometimes circumstances (typically server deployment) where it is more convenient to have IP addresses allocated sequentially, starting from the lowest available address, and setting this flag enables this mode. Note that in the sequential mode, clients which allow a lease to expire are much more likely to move IP address; for this reason it should not be generally used.

--dhcp-ignore-clid

Dnsmasq is reading 'client identifier' (RFC 2131) option sent by clients (if available) to identify clients. This allow to serve same IP address for a host using several interfaces. Use this option to disable 'client identifier' reading, i.e. to always identify a host using the MAC address.

--pxe-service=[tag:<tag>,]<CSA>,<menu text>[,<basename>|<bootservicetype>][,<server address>|<server_name>]

Most uses of PXE boot-ROMS simply allow the PXE system to obtain an IP address and then download the file specified by **--dhcp-boot** and execute it. However the PXE system is capable of more complex functions when supported by a suitable DHCP server.

This specifies a boot option which may appear in a PXE boot menu. <CSA> is client system type, only services of the correct type will appear in a menu. The known types are x86PC, PC98, IA64_EFI, Alpha, Arc_x86, Intel_Lean_Client, IA32_EFI, x86-64_EFI, Xscale_EFI, BC_EFI, ARM32_EFI and ARM64_EFI; an integer may be used for other types. The parameter after the menu text may be a file name, in which case dnsmasq acts as a boot server and directs the PXE client to download the file by TFTP, either from itself (**--enable-tftp** must be set for this to work) or another TFTP server if the final server address/name is given. Note that the "layer" suffix (normally ".0") is supplied by PXE, and need not be added to the basename. Alternatively, the basename may be a filename, complete with suffix, in which case no layer suffix is added. If an integer boot service type, rather than a basename is given, then the PXE client will search for a suitable boot service for that type on the network. This search may be done by broadcast, or direct to a server if its IP address/name is provided. If no boot service type or filename is provided (or a boot service type of 0 is specified) then the menu entry will abort the net boot procedure and continue booting from local media. The server address can be given as a domain name which is looked up in /etc/hosts. This name can be associated in /etc/hosts with multiple IP addresses, which are used round-robin.

--pxe-prompt=[tag:<tag>,]<prompt>[,<timeout>]

Setting this provides a prompt to be displayed after PXE boot. If the timeout is given then after the timeout has elapsed with no keyboard input, the first available menu option will be automatically executed. If the timeout is zero then the first available menu item will be executed immediately. If **--pxe-prompt** is omitted the system will wait for user input if there are multiple items in the menu, but boot immediately if there is only one. See **--pxe-service** for details of menu items.

Dnsmasq supports PXE "proxy-DHCP", in this case another DHCP server on the network is responsible for allocating IP addresses, and dnsmasq simply provides the information given in **--pxe-prompt** and **--pxe-service** to allow netbooting. This mode is enabled using the **proxy** keyword in **--dhcp-range.**

--dhcp-pxe-vendor=<vendor>[,...]
According to UEFI and PXE specifications, DHCP packets between PXE clients and proxy PXE servers should have *PXEClient* in their vendor-class field. However, the firmware of computers from a few vendors is customized to carry a different identifier in that field. This option is used to consider such identifiers valid for identifying PXE clients. For instance

--dhcp-pxe-vendor=PXEClient,HW-Client

will enable dnsmasq to also provide proxy PXE service to those PXE clients with *HW-Client* in as their identifier.

−X, --dhcp-lease-max=<number>
Limits dnsmasq to the specified maximum number of DHCP leases. The default is 1000. This limit is to prevent DoS attacks from hosts which create thousands of leases and use lots of memory in the dnsmasq process.

−K, --dhcp-authoritative
Should be set when dnsmasq is definitely the only DHCP server on a network. For DHCPv4, it changes the behaviour from strict RFC compliance so that DHCP requests on unknown leases from unknown hosts are not ignored. This allows new hosts to get a lease without a tedious timeout under all circumstances. It also allows dnsmasq to rebuild its lease database without each client needing to reacquire a lease, if the database is lost. For DHCPv6 it sets the priority in replies to 255 (the maximum) instead of 0 (the minimum).

--dhcp-rapid-commit
Enable DHCPv4 Rapid Commit Option specified in RFC 4039. When enabled, dnsmasq will respond to a DHCPDISCOVER message including a Rapid Commit option with a DHCPACK including a Rapid Commit option and fully committed address and configuration information. Should only be enabled if either the server is the only server for the subnet, or multiple servers are present and they each commit a binding for all clients.

--dhcp-alternate-port[=<server port>[,<client port>]]
(IPv4 only) Change the ports used for DHCP from the default. If this option is given alone, without arguments, it changes the ports used for DHCP from 67 and 68 to 1067 and 1068. If a single argument is given, that port number is used for the server and the port number plus one used for the client. Finally, two port numbers allows arbitrary specification of both server and client ports for DHCP.

−3, --bootp-dynamic[=<network-id>[,<network-id>]]
(IPv4 only) Enable dynamic allocation of IP addresses to BOOTP clients. Use this with care, since each address allocated to a BOOTP client is leased forever, and therefore becomes permanently unavailable for re-use by other hosts. if this is given without tags, then it unconditionally enables dynamic allocation. With tags, only when the tags are all set. It may be repeated with different tag sets.

−5, --no-ping
(IPv4 only) By default, the DHCP server will attempt to ensure that an address is not in use before allocating it to a host. It does this by sending an ICMP echo request (aka "ping") to the address in question. If it gets a reply, then the address must already be in use, and another is tried. This flag disables this check. Use with caution.

--log-dhcp
Extra logging for DHCP: log all the options sent to DHCP clients and the tags used to determine them.

--quiet-dhcp, --quiet-dhcp6, --quiet-ra, --quiet-tftp
Suppress logging of the routine operation of these protocols. Errors and problems will still be logged. **--quiet-tftp** does not consider file not found to be an error. **--quiet-dhcp** and quiet-dhcp6 are over-ridden by **--log-dhcp**.

−l, --dhcp-leasefile=<path>
Use the specified file to store DHCP lease information.

--dhcp-duid=<enterprise-id>,<uid>
(IPv6 only) Specify the server persistent UID which the DHCPv6 server will use. This option is not normally required as dnsmasq creates a DUID automatically when it is first needed. When given, this option provides dnsmasq the data required to create a DUID-EN type DUID. Note that once set, the DUID is stored in the lease database, so to change between DUID-EN and automatically created DUIDs or vice-versa, the lease database must be re-initialised. The enterprise-id is assigned by IANA, and the uid is a string of hex octets unique to a particular device.

−6 --dhcp-script=<path>
Whenever a new DHCP lease is created, or an old one destroyed, or a TFTP file transfer completes, the executable specified by this option is run. <path> must be an absolute pathname, no PATH search occurs. The arguments to the process are "add", "old" or "del", the MAC address of the host (or DUID for IPv6) , the IP address, and the hostname, if known. "add" means a lease has been created, "del" means it has been destroyed, "old" is a notification of an existing lease when dnsmasq starts or a change to MAC address or hostname of an existing lease (also, lease length or expiry and client-id, if **--leasefile-ro** is set and lease expiry if **--script-on-renewal** is set). If the MAC address is from a network type other than ethernet, it will have the network type prepended, eg "06-01:23:45:67:89:ab" for token ring. The process is run as root (assuming that dnsmasq was originally run as root) even if dnsmasq is configured to change UID to an unprivileged user.

The environment is inherited from the invoker of dnsmasq, with some or all of the following variables added

For both IPv4 and IPv6:

DNSMASQ_DOMAIN if the fully-qualified domain name of the host is known, this is set to the domain part. (Note that the hostname passed to the script as an argument is never fully-qualified.)

If the client provides a hostname, DNSMASQ_SUPPLIED_HOSTNAME

If the client provides user-classes, DNSMASQ_USER_CLASS0..DNSMASQ_USER_CLASSn

If dnsmasq was compiled with HAVE_BROKEN_RTC, then the length of the lease (in seconds) is stored in DNSMASQ_LEASE_LENGTH, otherwise the time of lease expiry is stored in DNSMASQ_LEASE_EXPIRES. The number of seconds until lease expiry is always stored in DNSMASQ_TIME_REMAINING.

If a lease used to have a hostname, which is removed, an "old" event is generated with the new state of the lease, ie no name, and the former name is provided in the environment variable DNSMASQ_OLD_HOSTNAME.

DNSMASQ_INTERFACE stores the name of the interface on which the request arrived; this is not set for "old" actions when dnsmasq restarts.

DNSMASQ_RELAY_ADDRESS is set if the client used a DHCP relay to contact dnsmasq and the IP address of the relay is known.

DNSMASQ_TAGS contains all the tags set during the DHCP transaction, separated by spaces.

DNSMASQ_LOG_DHCP is set if **--log-dhcp** is in effect.

For IPv4 only:

DNSMASQ_CLIENT_ID if the host provided a client-id.

DNSMASQ_CIRCUIT_ID, DNSMASQ_SUBSCRIBER_ID, DNSMASQ_REMOTE_ID if a DHCP relay-agent added any of these options.

If the client provides vendor-class, DNSMASQ_VENDOR_CLASS.

DNSMASQ_REQUESTED_OPTIONS a string containing the decimal values in the Parameter Request List option, comma separated, if the parameter request list option is provided by the client.

For IPv6 only:

If the client provides vendor-class, DNSMASQ_VENDOR_CLASS_ID, containing the IANA enterprise id for the class, and DNSMASQ_VENDOR_CLASS0..DNSMASQ_VENDOR_CLASSn for the data.

DNSMASQ_SERVER_DUID containing the DUID of the server: this is the same for every call to the script.

DNSMASQ_IAID containing the IAID for the lease. If the lease is a temporary allocation, this is prefixed to 'T'.

DNSMASQ_MAC containing the MAC address of the client, if known.

Note that the supplied hostname, vendorclass and userclass data is only supplied for "add" actions or "old" actions when a host resumes an existing lease, since these data are not held in dnsmasq's lease database.

All file descriptors are closed except stdin, which is open to /dev/null, and stdout and stderr which capture output for logging by dnsmasq. (In debug mode, stdio, stdout and stderr file are left as those inherited from the invoker of dnsmasq).

The script is not invoked concurrently: at most one instance of the script is ever running (dnsmasq waits for an instance of script to exit before running the next). Changes to the lease database are which require the script to be invoked are queued awaiting exit of a running instance. If this queueing allows multiple state changes occur to a single lease before the script can be run then earlier states are discarded and the current state of that lease is reflected when the script finally runs.

At dnsmasq startup, the script will be invoked for all existing leases as they are read from the lease file. Expired leases will be called with "del" and others with "old". When dnsmasq receives a HUP signal, the script will be invoked for existing leases with an "old" event.

There are four further actions which may appear as the first argument to the script, "init", "arp-add", "arp-del" and "tftp". More may be added in the future, so scripts should be written to ignore unknown actions. "init" is described below in **--leasefile-ro** The "tftp" action is invoked when a TFTP file transfer completes: the arguments are the file size in bytes, the address to which the file was sent, and the complete pathname of the file.

The "arp-add" and "arp-del" actions are only called if enabled with **--script-arp** They are are

supplied with a MAC address and IP address as arguments. "arp-add" indicates the arrival of a new entry in the ARP or neighbour table, and "arp-del" indicates the deletion of same.

--dhcp-luascript=<path>

Specify a script written in Lua, to be run when leases are created, destroyed or changed. To use this option, dnsmasq must be compiled with the correct support. The Lua interpreter is initialised once, when dnsmasq starts, so that global variables persist between lease events. The Lua code must define a **lease** function, and may provide **init** and **shutdown** functions, which are called, without arguments when dnsmasq starts up and terminates. It may also provide a **tftp** function.

The **lease** function receives the information detailed in **--dhcp-script.** It gets two arguments, firstly the action, which is a string containing, "add", "old" or "del", and secondly a table of tag value pairs. The tags mostly correspond to the environment variables detailed above, for instance the tag "domain" holds the same data as the environment variable DNSMASQ_DOMAIN. There are a few extra tags which hold the data supplied as arguments to **--dhcp-script.** These are **mac_address, ip_address** and **hostname** for IPv4, and **client_duid, ip_address** and **hostname** for IPv6.

The **tftp** function is called in the same way as the lease function, and the table holds the tags **destination_address, file_name** and **file_size.**

The **arp** and **arp-old** functions are called only when enabled with **--script-arp** and have a table which holds the tags **mac_address** and **client_address.**

--dhcp-scriptuser

Specify the user as which to run the lease-change script or Lua script. This defaults to root, but can be changed to another user using this flag.

--script-arp

Enable the "arp" and "arp-old" functions in the **--dhcp-script** and **--dhcp-luascript**.

−9, --leasefile-ro

Completely suppress use of the lease database file. The file will not be created, read, or written. Change the way the lease-change script (if one is provided) is called, so that the lease database may be maintained in external storage by the script. In addition to the invocations given in **--dhcp-script** the lease-change script is called once, at dnsmasq startup, with the single argument "init". When called like this the script should write the saved state of the lease database, in dnsmasq leasefile format, to stdout and exit with zero exit code. Setting this option also forces the leasechange script to be called on changes to the client-id and lease length and expiry time.

--script-on-renewal

Call the DHCP script when the lease expiry time changes, for instance when the lease is renewed.

--bridge-interface=<interface>,<alias>[,<alias>]

Treat DHCP (v4 and v6) requests and IPv6 Router Solicit packets arriving at any of the <alias> interfaces as if they had arrived at <interface>. This option allows dnsmasq to provide DHCP and RA service over unaddressed and unbridged Ethernet interfaces, e.g. on an OpenStack compute host where each such interface is a TAP interface to a VM, or as in "old style bridging" on BSD platforms. A trailing '*' wildcard can be used in each <alias>.

It is permissible to add more than one alias using more than one **--bridge-interface** option since **--bridge-interface=int1,alias1,alias2** is exactly equivalent to **--bridge-interface=int1,alias1 --bridge-interface=int1,alias2**

--shared-network=<interface>,<addr>
--shared-network=<addr>,<addr>

The DHCP server determines which DHCP ranges are useable for allocating an address to a DHCP client based on the network from which the DHCP request arrives, and the IP configuration

of the server's interface on that network. The shared-network option extends the available subnets (and therefore DHCP ranges) beyond the subnets configured on the arrival interface.

The first argument is either the name of an interface, or an address that is configured on a local interface, and the second argument is an address which defines another subnet on which addresses can be allocated.

To be useful, there must be a suitable dhcp-range which allows address allocation on this subnet and this dhcp-range MUST include the netmask.

Using shared-network also needs extra consideration of routing. Dnsmasq does not have the usual information that it uses to determine the default route, so the default route option (or other routing) MUST be configured manually. The client must have a route to the server: if the two-address form of shared-network is used, this needs to be to the first specified address. If the interface,address form is used, there must be a route to all of the addresses configured on the interface.

The two-address form of shared-network is also usable with a DHCP relay: the first address is the address of the relay and the second, as before, specifies an extra subnet which addresses may be allocated from.

−s, --domain=<domain>[,<address range>[,local]]

Specifies DNS domains for the DHCP server. Domains may be be given unconditionally (without the IP range) or for limited IP ranges. This has two effects; firstly it causes the DHCP server to return the domain to any hosts which request it, and secondly it sets the domain which it is legal for DHCP-configured hosts to claim. The intention is to constrain hostnames so that an untrusted host on the LAN cannot advertise its name via DHCP as e.g. "microsoft.com" and capture traffic not meant for it. If no domain suffix is specified, then any DHCP hostname with a domain part (ie with a period) will be disallowed and logged. If suffix is specified, then hostnames with a domain part are allowed, provided the domain part matches the suffix. In addition, when a suffix is set then hostnames without a domain part have the suffix added as an optional domain part. Eg on my network I can set **--domain=thekelleys.org.uk** and have a machine whose DHCP hostname is "laptop". The IP address for that machine is available from **dnsmasq** both as "laptop" and "laptop.thekelleys.org.uk". If the domain is given as "#" then the domain is read from the first "search" directive in /etc/resolv.conf (or equivalent).

The address range can be of the form <ip address>,<ip address> or <ip address>/<netmask> or just a single <ip address>. See **--dhcp-fqdn** which can change the behaviour of dnsmasq with domains.

If the address range is given as ip-address/network-size, then a additional flag "local" may be supplied which has the effect of adding **--local** declarations for forward and reverse DNS queries. Eg. **--domain=thekelleys.org.uk,192.168.0.0/24,local** is identical to **--domain=thekelleys.org.uk,192.168.0.0/24 --local=/thekelleys.org.uk/ --local=/0.168.192.in-addr.arpa/** The network size must be 8, 16 or 24 for this to be legal.

--dhcp-fqdn

In the default mode, dnsmasq inserts the unqualified names of DHCP clients into the DNS. For this reason, the names must be unique, even if two clients which have the same name are in different domains. If a second DHCP client appears which has the same name as an existing client, the name is transferred to the new client. If **--dhcp-fqdn** is set, this behaviour changes: the unqualified name is no longer put in the DNS, only the qualified name. Two DHCP clients with the same name may both keep the name, provided that the domain part is different (ie the fully qualified names differ.) To ensure that all names have a domain part, there must be at least **--domain** without an

address specified when **--dhcp-fqdn** is set.

--dhcp-client-update
Normally, when giving a DHCP lease, dnsmasq sets flags in the FQDN option to tell the client not to attempt a DDNS update with its name and IP address. This is because the name-IP pair is automatically added into dnsmasq's DNS view. This flag suppresses that behaviour, this is useful, for instance, to allow Windows clients to update Active Directory servers. See RFC 4702 for details.

--enable-ra
Enable dnsmasq's IPv6 Router Advertisement feature. DHCPv6 doesn't handle complete network configuration in the same way as DHCPv4. Router discovery and (possibly) prefix discovery for autonomous address creation are handled by a different protocol. When DHCP is in use, only a subset of this is needed, and dnsmasq can handle it, using existing DHCP configuration to provide most data. When RA is enabled, dnsmasq will advertise a prefix for each **--dhcp-range**, with default router as the relevant link-local address on the machine running dnsmasq. By default, the "managed address" bits are set, and the "use SLAAC" bit is reset. This can be changed for individual subnets with the mode keywords described in **--dhcp-range.** RFC6106 DNS parameters are included in the advertisements. By default, the relevant link-local address of the machine running dnsmasq is sent as recursive DNS server. If provided, the DHCPv6 options dns-server and domain-search are used for the DNS server (RDNSS) and the domain search list (DNSSL).

--ra-param=<interface>,[mtu:<integer>|<interface>|off,][high,|low,]<ra-interval>[,<router lifetime>]
Set non-default values for router advertisements sent via an interface. The priority field for the router may be altered from the default of medium with eg **--ra-param=eth0,high.** The interval between router advertisements may be set (in seconds) with **--ra-param=eth0,60.** The lifetime of the route may be changed or set to zero, which allows a router to advertise prefixes but not a route via itself. **--ra-param=eth0,0,0** (A v alue of zero for the interval means the default value.) All four parameters may be set at once. **--ra-param=eth0,mtu:1280,low,60,1200**

The interface field may include a wildcard.

The mtu: parameter may be an arbitrary interface name, in which case the MTU value for that interface is used. This is useful for (eg) advertising the MTU of a WAN interface on the other interfaces of a router.

--dhcp-reply-delay=[tag:<tag>,]<integer>
Delays sending DHCPOFFER and PROXYDHCP replies for at least the specified number of seconds. This can be used as workaround for bugs in PXE boot firmware that does not function properly when receiving an instant reply. This option takes into account the time already spent waiting (e.g. performing ping check) if any.

--enable-tftp[=<interface>[,<interface>]]
Enable the TFTP server function. This is deliberately limited to that needed to net-boot a client. Only reading is allowed; the tsize and blksize extensions are supported (tsize is only supported in octet mode). Without an argument, the TFTP service is provided to the same set of interfaces as DHCP service. If the list of interfaces is provided, that defines which interfaces receive TFTP service.

--tftp-root=<directory>[,<interface>]
Look for files to transfer using TFTP relative to the given directory. When this is set, TFTP paths which include ".." are rejected, to stop clients getting outside the specified root. Absolute paths (starting with /) are allowed, but they must be within the tftp-root. If the optional interface argument is given, the directory is only used for TFTP requests via that interface.

--tftp-no-fail
Do not abort startup if specified tftp root directories are inaccessible.

--tftp-unique-root[=ip|mac]
Add the IP or hardware address of the TFTP client as a path component on the end of the TFTP-root. Only valid if a **--tftp-root** is set and the directory exists. Defaults to adding IP address (in standard dotted-quad format). For instance, if **--tftp-root** is "/tftp" and client 1.2.3.4 requests file "myfile" then the effective path will be "/tftp/1.2.3.4/myfile" if /tftp/1.2.3.4 exists or /tftp/myfile otherwise. When "=mac" is specified it will append the MAC address instead, using lowercase zero padded digits separated by dashes, e.g.: 01-02-03-04-aa-bb Note that resolving MAC addresses is only possible if the client is in the local network or obtained a DHCP lease from us.

--tftp-secure
Enable TFTP secure mode: without this, any file which is readable by the dnsmasq process under normal unix access-control rules is available via TFTP. When the **--tftp-secure** flag is given, only files owned by the user running the dnsmasq process are accessible. If dnsmasq is being run as root, different rules apply: **--tftp-secure** has no effect, but only files which have the world-readable bit set are accessible. It is not recommended to run dnsmasq as root with TFTP enabled, and certainly not without specifying **--tftp-root**. Doing so can expose any world-readable file on the server to any host on the net.

--tftp-lowercase
Convert filenames in TFTP requests to all lowercase. This is useful for requests from Windows machines, which have case-insensitive filesystems and tend to play fast-and-loose with case in filenames. Note that dnsmasq's tftp server always converts "\" to "/" in filenames.

--tftp-max=<connections>
Set the maximum number of concurrent TFTP connections allowed. This defaults to 50. When serving a large number of TFTP connections, per-process file descriptor limits may be encountered. Dnsmasq needs one file descriptor for each concurrent TFTP connection and one file descriptor per unique file (plus a few others). So serving the same file simultaneously to n clients will use require about n + 10 file descriptors, serving different files simultaneously to n clients will require about (2*n) + 10 descriptors. If **--tftp-port-range** is given, that can affect the number of concurrent connections.

--tftp-mtu=<mtu size>
Use size as the ceiling of the MTU supported by the intervening network when negotiating TFTP blocksize, overriding the MTU setting of the local interface if it is larger.

--tftp-no-blocksize
Stop the TFTP server from negotiating the "blocksize" option with a client. Some buggy clients request this option but then behave badly when it is granted.

--tftp-port-range=<start>,<end>
A TFTP server listens on a well-known port (69) for connection initiation, but it also uses a dynamically-allocated port for each connection. Normally these are allocated by the OS, but this option specifies a range of ports for use by TFTP transfers. This can be useful when TFTP has to traverse a firewall. The start of the range cannot be lower than 1025 unless dnsmasq is running as root. The number of concurrent TFTP connections is limited by the size of the port range.

--tftp-single-port
Run in a mode where the TFTP server uses ONLY the well-known port (69) for its end of the TFTP transfer. This allows TFTP to work when there in NAT is the path between client and server. Note that this is not strictly compliant with the RFCs specifying the TFTP protocol: use at your

own risk.

−C, --conf-file=<file>
Specify a configuration file. The presence of this option stops dnsmasq from reading the default configuration file (normally /etc/dnsmasq.conf). Multiple files may be specified by repeating the option either on the command line or in configuration files. A filename of "-" causes dnsmasq to read configuration from stdin.

−7, --conf-dir=<directory>[,<file-extension>.....],
Read all the files in the given directory as configuration files. If extension(s) are given, any files which end in those extensions are skipped. Any files whose names end in ˜ or start with . or start and end with # are always skipped. If the extension starts with * then only files which have that extension are loaded. So **--conf-dir=/path/to/dir,*.conf** loads all files with the suffix .conf in /path/to/dir. This flag may be given on the command line or in a configuration file. If giving it on the command line, be sure to escape * characters. Files are loaded in alphabetical order of filename.

--servers-file=<file>
A special case of **--conf-file** which differs in two respects. Firstly, only **--server** and **--rev-server** are allowed in the configuration file included. Secondly, the file is re-read and the configuration therein is updated when dnsmasq receives SIGHUP.

CONFIG FILE

At startup, dnsmasq reads */etc/dnsmasq.conf*, if it exists. (On FreeBSD, the file is */usr/local/etc/dnsmasq.conf*) (but see the **−-conf-file** and **−-conf-dir** options.) The format of this file consists of one option per line, exactly as the long options detailed in the OPTIONS section but without the leading "--". Lines starting with # are comments and ignored. For options which may only be specified once, the configuration file overrides the command line. Quoting is allowed in a config file: between " quotes the special meanings of ,:. and # are removed and the following escapes are allowed: \\ \" \t \e \b \r and \n. The later corresponding to tab, escape, backspace, return and newline.

NOTES

When it receives a SIGHUP, **dnsmasq** clears its cache and then re-loads */etc/hosts* and */etc/ethers* and any file given by **--dhcp-hostsfile**, **--dhcp-hostsdir**, **--dhcp-optsfile**, **--dhcp-optsdir**, **--addn-hosts** or **--hostsdir**. The DHCP lease change script is called for all existing DHCP leases. If **--no-poll** is set SIGHUP also re-reads */etc/resolv.conf*. SIGHUP does NOT re-read the configuration file.

When it receives a SIGUSR1, **dnsmasq** writes statistics to the system log. It writes the cache size, the number of names which have had to removed from the cache before they expired in order to make room for new names and the total number of names that have been inserted into the cache. The number of cache hits and misses and the number of authoritative queries answered are also given. For each upstream server it gives the number of queries sent, and the number which resulted in an error. In **--no-daemon** mode or when full logging is enabled (**--log-queries**), a complete dump of the contents of the cache is made.

The cache statistics are also available in the DNS as answers to queries of class CHAOS and type TXT in domain bind. The domain names are cachesize.bind, insertions.bind, evictions.bind, misses.bind, hits.bind, auth.bind and servers.bind. An example command to query this, using the **dig** utility would be

dig +short chaos txt cachesize.bind

When it receives SIGUSR2 and it is logging direct to a file (see **--log-facility**) **dnsmasq** will close and reopen the log file. Note that during this operation, dnsmasq will not be running as root. When it first creates the logfile dnsmasq changes the ownership of the file to the non-root user it will run as. Logrotate should be

configured to create a new log file with the ownership which matches the existing one before sending SIGUSR2. If TCP DNS queries are in progress, the old logfile will remain open in child processes which are handling TCP queries and may continue to be written. There is a limit of 150 seconds, after which all existing TCP processes will have expired: for this reason, it is not wise to configure logfile compression for logfiles which have just been rotated. Using logrotate, the required options are **create** and **delaycompress.**

Dnsmasq is a DNS query forwarder: it is not capable of recursively answering arbitrary queries starting from the root servers but forwards such queries to a fully recursive upstream DNS server which is typically provided by an ISP. By default, dnsmasq reads */etc/resolv.conf* to discover the IP addresses of the upstream nameservers it should use, since the information is typically stored there. Unless **--no-poll** is used, **dnsmasq** checks the modification time of */etc/resolv.conf* (or equivalent if **−-resolv-file** is used) and re-reads it if it changes. This allows the DNS servers to be set dynamically by PPP or DHCP since both protocols provide the information. Absence of */etc/resolv.conf* is not an error since it may not have been created before a PPP connection exists. Dnsmasq simply keeps checking in case */etc/resolv.conf* is created at any time. Dnsmasq can be told to parse more than one resolv.conf file. This is useful on a laptop, where both PPP and DHCP may be used: dnsmasq can be set to poll both */etc/ppp/resolv.conf* and */etc/dhcpc/resolv.conf* and will use the contents of whichever changed last, giving automatic switching between DNS servers.

Upstream servers may also be specified on the command line or in the configuration file. These server specifications optionally take a domain name which tells dnsmasq to use that server only to find names in that particular domain.

In order to configure dnsmasq to act as cache for the host on which it is running, put "nameserver 127.0.0.1" in */etc/resolv.conf* to force local processes to send queries to dnsmasq. Then either specify the upstream servers directly to dnsmasq using **−-server** options or put their addresses real in another file, say */etc/resolv.dnsmasq* and run dnsmasq with the **−-resolv-file /etc/resolv.dnsmasq** option. This second technique allows for dynamic update of the server addresses by PPP or DHCP.

Addresses in /etc/hosts will "shadow" different addresses for the same names in the upstream DNS, so "mycompany.com 1.2.3.4" in /etc/hosts will ensure that queries for "mycompany.com" always return 1.2.3.4 even if queries in the upstream DNS would otherwise return a different address. There is one exception to this: if the upstream DNS contains a CNAME which points to a shadowed name, then looking up the CNAME through dnsmasq will result in the unshadowed address associated with the target of the CNAME. To work around this, add the CNAME to /etc/hosts so that the CNAME is shadowed too.

The tag system works as follows: For each DHCP request, dnsmasq collects a set of valid tags from active configuration lines which include set:<tag>, including one from the **--dhcp-range** used to allocate the address, one from any matching **--dhcp-host** (and "known" or "known-othernet" if a **--dhcp-host** matches) The tag "bootp" is set for BOOTP requests, and a tag whose name is the name of the interface on which the request arrived is also set.

Any configuration lines which include one or more tag:<tag> constructs will only be valid if all that tags are matched in the set derived above. Typically this is **--dhcp-option**. **--dhcp-option** which has tags will be used in preference to an untagged **--dhcp-option,** provided that _all_ the tags match somewhere in the set collected as described above. The prefix '!' on a tag means 'not' so **--dhcp-option=tag:!purple,3,1.2.3.4** sends the option when the tag purple is not in the set of valid tags. (If using this in a command line rather than a configuration file, be sure to escape !, which is a shell metacharacter)

When selecting **--dhcp-options**, a tag from **--dhcp-range** is second class relative to other tags, to make it easy to override options for individual hosts, so **--dhcp-range=set:interface1,..... --dhcp-host=set:myhost,..... --dhcp-option=tag:interface1,option:nis-domain,"domain1" --dhcp-**

option=tag:myhost,option:nis-domain,"domain2" will set the NIS-domain to domain1 for hosts in the range, but override that to domain2 for a particular host.

Note that for **--dhcp-range** both tag:<tag> and set:<tag> are allowed, to both select the range in use based on (eg) **--dhcp-host**, and to affect the options sent, based on the range selected.

This system evolved from an earlier, more limited one and for backward compatibility "net:" may be used instead of "tag:" and "set:" may be omitted. (Except in **--dhcp-host,** where "net:" may be used instead of "set:".) For the same reason, '#' may be used instead of '!' to indicate NOT.

The DHCP server in dnsmasq will function as a BOOTP server also, provided that the MAC address and IP address for clients are given, either using **--dhcp-host** configurations or in */etc/ethers* , and a **--dhcp-range** configuration option is present to activate the DHCP server on a particular network. (Setting **--bootp-dynamic** removes the need for static address mappings.) The filename parameter in a BOOTP request is used as a tag, as is the tag "bootp", allowing some control over the options returned to different classes of hosts.

AUTHORITATIVE CONFIGURATION

Configuring dnsmasq to act as an authoritative DNS server is complicated by the fact that it involves configuration of external DNS servers to provide delegation. We will walk through three scenarios of increasing complexity. Prerequisites for all of these scenarios are a globally accessible IP address, an A or AAAA record pointing to that address, and an external DNS server capable of doing delegation of the zone in question. For the first part of this explanation, we will call the A (or AAAA) record for the globally accessible address server.example.com, and the zone for which dnsmasq is authoritative our.zone.com.

The simplest configuration consists of two lines of dnsmasq configuration; something like

--auth-server=server.example.com,eth0
--auth-zone=our.zone.com,1.2.3.0/24

and two records in the external DNS

server.example.com A 192.0.43.10
our.zone.com NS server.example.com

eth0 is the external network interface on which dnsmasq is listening, and has (globally accessible) address 192.0.43.10.

Note that the external IP address may well be dynamic (ie assigned from an ISP by DHCP or PPP) If so, the A record must be linked to this dynamic assignment by one of the usual dynamic-DNS systems.

A more complex, but practically useful configuration has the address record for the globally accessible IP address residing in the authoritative zone which dnsmasq is serving, typically at the root. Now we have

--auth-server=our.zone.com,eth0
--auth-zone=our.zone.com,1.2.3.0/24

our.zone.com A 1.2.3.4
our.zone.com NS our.zone.com

The A record for our.zone.com has now become a glue record, it solves the chicken-and-egg problem of finding the IP address of the nameserver for our.zone.com when the A record is within that zone. Note that this is the only role of this record: as dnsmasq is now authoritative from our.zone.com it too must provide

this record. If the external address is static, this can be done with an **/etc/hosts** entry or **--host-record.**

--auth-server=our.zone.com,eth0
--host-record=our.zone.com,1.2.3.4
--auth-zone=our.zone.com,1.2.3.0/24

If the external address is dynamic, the address associated with our.zone.com must be derived from the address of the relevant interface. This is done using **--interface-name** Something like:

--auth-server=our.zone.com,eth0
--interface-name=our.zone.com,eth0
--auth-zone=our.zone.com,1.2.3.0/24,eth0

(The "eth0" argument in **--auth-zone** adds the subnet containing eth0's dynamic address to the zone, so that the **--interface-name** returns the address in outside queries.)

Our final configuration builds on that above, but also adds a secondary DNS server. This is another DNS server which learns the DNS data for the zone by doing zones transfer, and acts as a backup should the primary server become inaccessible. The configuration of the secondary is beyond the scope of this man-page, but the extra configuration of dnsmasq is simple:

--auth-sec-servers=secondary.myisp.com

and

our.zone.com NS secondary.myisp.com

Adding auth-sec-servers enables zone transfer in dnsmasq, to allow the secondary to collect the DNS data. If you wish to restrict this data to particular hosts then

--auth-peer=<IP address of secondary>

will do so.

Dnsmasq acts as an authoritative server for in-addr.arpa and ip6.arpa domains associated with the subnets given in **--auth-zone** declarations, so reverse (address to name) lookups can be simply configured with a suitable NS record, for instance in this example, where we allow 1.2.3.0/24 addresses.

3.2.1.in-addr.arpa NS our.zone.com

Note that at present, reverse (in-addr.arpa and ip6.arpa) zones are not available in zone transfers, so there is no point arranging secondary servers for reverse lookups.

When dnsmasq is configured to act as an authoritative server, the following data is used to populate the authoritative zone.

--mx-host, --srv-host, --dns-rr, --txt-record, --naptr-record, --caa-record, as long as the record names are in the authoritative domain.

--synth-domain as long as the domain is in the authoritative zone and, for reverse (PTR) queries, the address is in the relevant subnet.

--cname as long as the record name is in the authoritative domain. If the target of the CNAME is

unqualified, then it is qualified with the authoritative zone name. CNAME used in this way (only) may be wildcards, as in

--cname=*.example.com,default.example.com

IPv4 and IPv6 addresses from /etc/hosts (and **--addn-hosts**) and **--host-record** and **--interface-name** and **---dynamic-host** provided the address falls into one of the subnets specified in the **--auth-zone.**

Addresses of DHCP leases, provided the address falls into one of the subnets specified in the **--auth-zone.** (If constructed DHCP ranges are is use, which depend on the address dynamically assigned to an interface, then the form of **--auth-zone** which defines subnets by the dynamic address of an interface should be used to ensure this condition is met.)

In the default mode, where a DHCP lease has an unqualified name, and possibly a qualified name constructed using **--domain** then the name in the authoritative zone is constructed from the unqualified name and the zone's domain. This may or may not equal that specified by **--domain.** If **--dhcp-fqdn** is set, then the fully qualified names associated with DHCP leases are used, and must match the zone's domain.

EXIT CODES

0 - Dnsmasq successfully forked into the background, or terminated normally if backgrounding is not enabled.

1 - A problem with configuration was detected.

2 - A problem with network access occurred (address in use, attempt to use privileged ports without permission).

3 - A problem occurred with a filesystem operation (missing file/directory, permissions).

4 - Memory allocation failure.

5 - Other miscellaneous problem.

11 or greater - a non zero return code was received from the lease-script process "init" call. The exit code from dnsmasq is the script's exit code with 10 added.

LIMITS

The default values for resource limits in dnsmasq are generally conservative, and appropriate for embedded router type devices with slow processors and limited memory. On more capable hardware, it is possible to increase the limits, and handle many more clients. The following applies to dnsmasq-2.37: earlier versions did not scale as well.

Dnsmasq is capable of handling DNS and DHCP for at least a thousand clients. The DHCP lease times should not be very short (less than one hour). The value of **--dns-forward-max** can be increased: start with it equal to the number of clients and increase if DNS seems slow. Note that DNS performance depends too on the performance of the upstream nameservers. The size of the DNS cache may be increased: the hard limit is 10000 names and the default (150) is very low. Sending SIGUSR1 to dnsmasq makes it log information which is useful for tuning the cache size. See the **NOTES** section for details.

The built-in TFTP server is capable of many simultaneous file transfers: the absolute limit is related to the number of file-handles allowed to a process and the ability of the select() system call to cope with large numbers of file handles. If the limit is set too high using **--tftp-max** it will be scaled down and the actual limit logged at start-up. Note that more transfers are possible when the same file is being sent than when each transfer sends a different file.

It is possible to use dnsmasq to block Web advertising by using a list of known banner-ad servers, all resolving to 127.0.0.1 or 0.0.0.0, in **/etc/hosts** or an additional hosts file. The list can be very long, dnsmasq has been tested successfully with one million names. That size file needs a 1GHz processor and about 60Mb of RAM.

INTERNATIONALISATION

Dnsmasq can be compiled to support internationalisation. To do this, the make targets "all-i18n" and "install-i18n" should be used instead of the standard targets "all" and "install". When internationalisation is compiled in, dnsmasq will produce log messages in the local language and support internationalised domain names (IDN). Domain names in /etc/hosts, /etc/ethers and /etc/dnsmasq.conf which contain non-ASCII characters will be translated to the DNS-internal punycode representation. Note that dnsmasq determines both the language for messages and the assumed charset for configuration files from the LANG environment variable. This should be set to the system default value by the script which is responsible for starting dnsmasq. When editing the configuration files, be careful to do so using only the system-default locale and not user-specific one, since dnsmasq has no direct way of determining the charset in use, and must assume that it is the system default.

FILES

/etc/dnsmasq.conf

/usr/local/etc/dnsmasq.conf

/etc/resolv.conf /var/run/dnsmasq/resolv.conf /etc/ppp/resolv.conf /etc/dhcpc/resolv.conf

/etc/hosts

/etc/ethers

/var/lib/misc/dnsmasq.leases

/var/db/dnsmasq.leases

/var/run/dnsmasq.pid

SEE ALSO

hosts(5), **resolver**(5)

AUTHOR

This manual page was written by Simon Kelley <simon@thekelleys.org.uk>.