NAME

dhcp.conf − dynamic host configuration protocol configuration

SYNOPSIS

/etc/dhcp.conf

DESCRIPTION

The file **/etc/dhcp.conf** contains the configuration for the DHCP client/server program **dhcpd**. This text is a long summation of all the elements that can be found in this configuration file. For a more "just tell me what to do" approach see **boot**(8).

The syntax used is that of the common configuration file described in **configfile**(5).

To find information for a client we first need its IP address. Occasionally this IP address is already known (the special "INFORM" query), but normally we have to make a first pass through the configuration file for a **client** entry. If that fails then we use an IP address from the pool file. If we now have an IP address then the real information gathering can begin.

The DHCP daemon reads the configuration file from beginning to end and gathers all information that matches, and information from all macros that are mentioned within the elements that match. If we end up with DHCP information that includes at least a netmask definition, and is good for the network the request came in from, then it is returned to the client. If a DHCP tag is specified twice then the last one wins.

In the description below we use [and] to denote optional things, and | to show a choice between two things.

Client IDs can be either ordinary Ethernet addresses, that are treated as a seven octet string (01 followed by the Ethernet address), or any random octet string in hexadecimal.

IP addresses can be simply that, or host names. These host names are searched in **/etc/hosts** by **dhcpd** itself using a domain based prefix match, i.e. you can use "flotsam" for "flotsam.example.com", but not "alpha" for "alphabeta". Once the program decides to be a server it will also look up names normally in the DNS. If a host has more than one IP address then the address on the network the query was seen on is used.

Case isn't important in the configuration file, "client", "CLIENT" and "ClIeNt" are all treated the same.

Some elements may optionally name a macro or a curly braces enclosed parameter list of more elements. If the element matches then the data in the macro body or parameter list is gathered.

The following elements can be used:

client *client-ID* [**ip**#] *host* [*macro*|{*params*}];

> Defines a client with a given client ID that is to have the IP address denoted by *host* . On the first pass only the client ID is matched looking for an IP address that lies on the network the request came in on. On the information gathering pass both client ID and IP address must match. If a machine has the same Ethernet address on two or more interfaces then the IP address given out is the one on the same network as the request came in on. The optional interface name (**ip**#) must be used if the DHCP daemon is gathering data for itself at boot time to differentiate interfaces with the same ethernet addresses. This is only necessary under Minix-vmd when ethernets on different VLANs share the same physical ethernet. The interface name is only used for a machine's own networks, it ignored on entries for other hosts.

class *class-name ... macro*|{*params*};

> Includes the macro or parameters if one of the class names is matched. A host normally includes a class ID in its request. Minix and Minix-vmd use "Minix" as the class name. For Windows the class ID starts with "MSFT", and Solaris' starts with "SUNW". (Use **dhcpd −d3** to find out what the full IDs are exactly.) The class names are matched if a *class-name* is a prefix of the class ID sent by the client.

host *host-spec macro*|{*params*};

Includes the macro or parameters if the IP address of the client matches the host specification. This can either be an ordinary hostname, or a netblock in CIDR notation, e.g. 172.35.0.0/16. The example includes all IP addresses whose top 16 bits are the same as the top 16 bits of 172.35.0.0. Such a netblock automatically defines a netmask (255.255.0.0 in the example) if no netmask has been specified yet.

interface ip# *host* [*macro*|{*params*}];

Makes **dhcpd** set the IP address of interface **ip**# (where # is a number) to the IP address denoted by *host*. This element should only be used for interfaces that are not true Ethernets, and so do not have a unique Ethernet address that can be used for a client element. If the machine has at least one true Ethernet then all interface elements should be added to the parameter list of a host or client element for that Ethernet interface. This binds them to that machine and allows a single configuration file to be shared among machines. Especially a server should never have "free" interface elements. The macro or parameters are only evaluated if data is gathered for the given interface. (Note that they will be hidden by a client element for another interface.)

macro *macro-name*;

Include the parameter list of the macro named *macro-name* defined elsewhere. (This means that "host flotsam stuff" is just short for "host flotsam { macro stuff; }".)

macro *macro-name* {*params*};

Defines a macro with the given parameter list. Whenever the macro is used the parameter list is substituted instead. A macro can not be defined within another parameter list.

option [**ip**#] **server** [**inform**];
option [**ip**#] **relay** *host*;
option [**ip**#] **possessive**;
option [**ip**#] **hostname** *name*;

Makes **dhcpd** set special options for the interface that it is gathering data for, or the interface denoted by the optional **ip**# argument. The options are:

server [**inform**]
Be a DHCP server on the network connected to the interface. Add the word **inform** if DHCPINFORM requests must be answered for hosts we don't have an address for.

relay *host*
Be a DHCP relay to the indicated host.

possessive
Do not disable the interface if the DHCP lease expires. Useful if the DHCP server of the provider is unreliable, crashing a lot and causing the lease to expire. (Think twice before turning this option on. You have to be absolutely sure that it's the DHCP server that's the culprit and not a flaky network. You don't want an IP address conflict to be your fault.)

hostname *name*
Use the given name as our hostname in the DHCP queries. Some sites key on that bit of information instead of a client ID.

tag *number name type granularity max*;

Defines a DHCP tag for the given tag number and gives it a name. The name can be used in the configuration file to set tag values when gathering data. The *type* field can be one of **ascii**, **boolean**, **ip**, **number** or **octet** to specify the type of the tag as a simple string, a boolean, an IP address, a number, or a string of octet values. The *granularity* field specifies that that number of items must be given or a multiple thereof, unless the type is a number, then it is the size of the number (1, 2 or 4). The *max* field tells the maximum number of items that may be used with the tag, with 0 meaning "unlimited".

Three tags, the ones that Minix really cares about, have been predefined, and there are also a few pseudotags predefined for the static fields in a DHCP packet that one may want to set:

> tag ? siaddr ip 1 1;
> tag ? sname ascii 1 64;
> tag ? file ascii 1 128;
> tag 1 netmask ip 1 1;
> tag 3 gateway ip 1 0;
> tag 6 DNSserver ip 1 0;

The file **/usr/etc/dhcptags.conf** contains tag definitions for all standard DHCP tags. It is wise to include this file at the top of any DHCP configuration file.

no *tag-name*;

Removes a tag with the given name from the data gathered at this point. Useful if one host is different from all others, for instance if it doesn't need a gateway definition, because it happens to be the gateway.

ascii-tag string;

Adds an ASCII tag to the gathered data. The string can be a simple word, or a quoted string.

boolean-tag **false**|**true**;

Set a boolean tag to be false or true. (Encoded as a octet of value 0 or 1. Note that if you prefer to use 0 or 1 instead of false or true then you can define a boolean tag as a size 1 number instead.)

ip-tag host ...;

Sets a tag that needs one or more IP addresses. The host names are translated as usual. To make it easier to specify netmasks one can use a slash followed by a number, e.g. **netmask /27**, which is a handy alternative for **netmask 255.255.255.224**.

number-tag number ...;

Set a number tag.

octet-tag hexdigits;

Set an octet string tag. *Hexdigits* is a series of hexadecimal digits that are two by two used to set the octets.

EXAMPLE

As an example the DHCP configuration used by the author of this document is included. His network at home consists of a number of PCs, an ISDN router named rhone and a PC named saone serving as router/tunnel to/via a cable ISP. Both the rhone and the saone connect the home net to the network of the Vrije Universiteit, but the rhone is only active if the cable doesn't work.

The saone is a DHCP server, and one of the ordinary PCs is a backup DHCP server. Both use the same configuration file, which is added below, with extra commentary introduced by **##** at a deeper indent level:

```
include /usr/etc/dhcptags.conf;
```

With the help of the tag definitions we can use tags like "DHCPlease".

```
host 130.37.102.64/27 {
          DNSserver saone darask;
          host 130.37.102.88/29 { DHCPlease 259200; }
};
```

This defines the network 130.37.102.64/27, with netmask

255.255.255.224 (implicit from the network definition). The DNS servers for this net are saone and darask. A smaller subrange of addresses is used as an address pool on the saone, so a lease of 259200 seconds (3 days) is defined. The netmask is still /27, as set by the outer network definition.

```
host 130.37.102.248/30 {};
```

A network of two addresses for routing tests.

```
host saone {
            option server;
            option ip1 possessive;
            interface ip2 saone-net2;
            DNSserver 130.37.24.3 130.37.24.6;
};
```

With the network definitions done we turn our attention to the hosts. Saone is a DHCP server on its main interface. The second interface (**ip1**) is connected to the cable modem. It gets its address from the cableco's DHCP server, and if that server decides to go deaf then the saone keeps the interface up ("possessive") even if the lease expires. The pseudo IP interface **ip2** is set to the IP address of **saone-net2**, one side of the encrypted tunnel over the cable to a Minix-vmd box at the VU. The DNS servers specified override the default setting for the network, naming two external servers at the VU that know the world.

```
host darask {
            option server;
            DNSserver saone;
            class Minix { DNSserver saone 130.37.24.3 130.37.24.6; };
};
```

The darask is also a server, the backup for saone on the odd chance that it is unavailable. It uses saone and the external name servers, but only when it is running Minix. When running Windows it only uses saone.

```
client 0:1:1b:a:68:ce                              darask;
client 0:1:1b:a:77:23                              burask;
#lient 0:0:c0:b0:da:64                             burask;
client 8:0:5a:38:b2:f                              finiah;
client 0:0:c0:3a:12:10                             bardelask;
#lient 2:60:8c:ab:8a:6d                            bardelask;
client 0:a0:c5:20:9:6d                             rhone;
client 0:1:1b:a:4c:3b                              saone;
#lient 0:0:c0:fb:2:6a                              saone-net1;
```

Lastly the ethernet addresses of all the hosts are listed, so that they can be translated to IP addresses. The lines that are commented out are for extra network cards that are currently unused. The last is used to connect to the cable modem, so it's only here because it's nice to have the ethernet address written down somewhere.

The host names shown above are translated by DHCP using this **/etc/hosts**:

```
604800          %ttl
2419200         %stale

130.37.102.65   darask.kjb.upwind.org
130.37.102.66   burask.kjb.upwind.org
130.37.102.67   finiah.kjb.upwind.org
130.37.102.68   bardelask.kjb.upwind.org
130.37.102.69   roniah.kjb.upwind.org

130.37.102.70   saone.kjb.upwind.org
130.37.102.2    saone-net2.kjb.upwind.org

130.37.102.88   rhone.kjb.upwind.org
130.37.102.89   dyn89.kjb.upwind.org
130.37.102.90   dyn90.kjb.upwind.org
130.37.102.91   dyn91.kjb.upwind.org
130.37.102.92   dyn92.kjb.upwind.org
130.37.102.93   dyn93.kjb.upwind.org
130.37.102.94   dyn94.kjb.upwind.org

130.37.102.249  tst1.kjb.upwind.org
130.37.102.250  tst2.kjb.upwind.org
```

FILES

/usr/etc/dhcptags.conf

A supplied list of standard tag definitions as per RFC-1533. (Well, the tag numbers and their meaning are standard, the names are made up.)

SEE ALSO

RFC-2131, **RFC-1533**, **configfile**(5), **hosts**(5), **boot**(8), **dhcpd**(8).

NOTES

The amount of memory **dhcpd** needs increases with the size of configuration file. Minix can handle **dhcptags.conf** and a modest sized **dhcp.conf**. You have to increase the stack size to accommodate more. (No problem under Minix-vmd, of course.)

NOTES

Items that are only necessary for a certain host should only be specified for that host. Items for a whole network are best added to a netblock specification. Use class elements for a certain type of host, and macros for exceptions. Try to limit information as much as possibly to those hosts that need it. (Don't go overboard. A Minix machine won't be bothered by a few NetBIOS tags.)

DHCPINFORM requests should always be answered when being a server, but J. Random Sysadmin trying to diagnose problems doesn't like it when little Minix machines show up in a packet trace unexpectedly. It's best to be inconspicuous on a network you don't own.

BUGS

There are a few too many subtle mistakes one can make.

AUTHOR

Kees J. Bot <kjb@cs.vu.nl>