# Managing DHCP Server

This chapter describes how to set up some of the DHCP server parameters. Before clients can use DHCP for address assignment, you must add at least one scope to the server. This is described in Managing Scopes, Prefixes, Links, and Networks.

Cisco Prime Network Registrar failover protocol is designed to allow a backup DHCP server to take over for a main server if the main server is taken offline for any reason. To set up DHCP failover, read Managing DHCP Failover.



# Configuring DHCP Servers

When configuring a DHCP server, you must configure the server properties, policies, and associated DHCP options. Cisco Prime Network Registrar needs:

- The DHCP server IP address
- One or more scopes (see the Managing Scopes) and/or prefixes

## General Configuration Guidelines

Here are some guidelines to consider before configuring a DHCP server:

- **Separate the DHCP server from secondary DNS servers used for DNS updating**—To ensure that the DHCP server is not adversely affected during large zone transfers, it should run on a different cluster than your secondary DNS servers.
- **Lease times**—See the Guidelines for Lease Times.

# Configuring DHCP Server Interfaces

To configure the DHCP server, accept the Cisco Prime Network Registrar defaults or supply the data explicitly:

- **Network interface**—Ethernet card IP address, which must be static and not assigned by DHCP.
- **Subnet mask**—Identifies the interface network membership. The subnet mask is usually based on the network class of the interface address, in most cases 255.255.255.0.

By default, the DHCP server uses the operating system support to automatically enumerate the active interfaces on the machine and listens on all of them. You can also manually configure the server interface. You should statically configure all the IP addresses assigned to NIC cards on the machine where the DHCP server resides. The machine should not be a BOOTP or DHCP client.

**Note** Unless you have a specific need to restrict the interfaces used for DHCP, it is recommended that you do not configure specific DHCP Server interfaces. Allow the server to automatically discover the available interfaces.

## Local Advanced Web UI

**Step 1** From the **Operate** menu, choose **Manage Servers** under the **Servers** submenu to open the Manage Servers page.

**Step 2** Click **DHCP** in the Manager Servers pane to open the Edit Local DHCP Server page.

**Step 3** Click the **Network Interfaces** tab to view the available network interfaces that you can configure for the server. By default, the server uses all of them.

**Step 4** To configure an interface, click the **Edit** icon in the Configure column for the interface. This adds the interface to the Configured Interfaces table, where you can edit or delete it.

**Step 5** Clicking the name of the configured interface opens the Edit DHCP Server Network Interface page, where you can change the address and ports (in Expert mode) of the interface.

**Step 6** Click **Save** to save the changes.

**Step 7** Click **Revert** to return to the Manage Servers page.

## CLI Commands

Use **dhcp-interface** to manually control which network interface cards' IP addresses the DHCP server will listen on for DHCP clients. By default, the DHCP server automatically uses all your server network interfaces, so use this command to be more specific about which ones to use.

To troubleshoot and confirm validity of the configuration changes.

- Reload the DHCP server.
- Check the dhcp_startup_log and/or name_dhcp_1_log file.

For information on Log Settings, see Tuning the DHCP Server, on page 16.

# Defining Advanced Server Attributes

You can set advanced DHCP server attributes, including custom DHCP options.

To set up the DHCP server.

1. Configure a scope or prefix.
2. Reload the server.

# Related Topics

Setting Advanced DHCP Server Attributes, on page 3

Enabling BOOTP for Scopes, on page 36

Moving or Decommissioning BOOTP Clients, on page 37

Using Dynamic BOOTP, on page 37

BOOTP Relay, on page 38

# Setting Advanced DHCP Server Attributes

The table below describes the advanced DHCP server attributes that you can set in the local cluster web UI and CLI.

***Table 1: DHCP Advanced Attributes***

| Advanced Parameter | Action | Description |
|---|---|---|
| *max-dhcp-requests* | set/unset | Controls the number of buffers that the DHCP server allocates for receiving packets from DHCP clients and failover partners. If this setting is too large, a burst of DHCP activity can clog the server with requests that become stale before being processed. This results in an increasing processing load that can severely degrade performance as clients try to obtain a new lease, and affects the ability to handle bursts. A low buffer setting throttles requests and could affect server throughput. If the server runs out of buffers, packets are dropped.<br><br>A good rule or thumb is to increase the buffers if you expect a high load (in a steady state or when experiencing frequent stress times) or you have a fast multiprocessor system. |

| Advanced Parameter | Action | Description |
|---|---|---|
| | | In a nonfailover deployment, the default setting (500) is sufficient. In a failover deployment, you can increase it to 1000 if the DHCP logs indicate a consistently high number of request buffers. You should then also modify the number of DHCP responses (see the *max-dhcp-responses* parameter) to four times the request buffers.<br><br>When using LDAP client lookups, buffers should not exceed the LDAP lookup queue size defined by the total number of LDAP connections and the maximum number of requests allowed for each connection. Set the LDAP queue size to match the capacity of the LDAP server to service client lookups.<br><br>If the following logs messages occur frequently and are not related to short term traffic spikes (such as after a power recovery), you may want to consider increasing the value of the attribute:<br><br>`4493 DHCP ERROR "DHCP has used xx of its yy request buffers: the server is dropping a request."`<br>`4494 DHCP WARNING "DHCP has used xx of yy request packets. Requests will be ignored if no packet buffers are available."`<br>`5270 DHCP WARNING "DHCP has used xx of its yy request buffers: the server is congested -- will not keep the client last-transaction-time to within value but will keep it to within value seconds."`<br><br>Required. The default is 500. |

| Advanced Parameter | Action | Description |
|---|---|---|
| *max-dhcp-responses* | set/unset | Controls the number of response buffers that the DHCP server allocates for responding to DHCP clients and performing failover communication between DHCP partners.<br><br>In a non-failover deployment, the default setting of twice the number of request buffers is sufficient. In a failover deployment, you can increase this so that it is four times the number of request buffers. In general, increasing the number of response buffers is not harmful, while reducing it to below the previously recommended ratios might be harmful to server responsiveness.<br><br>If the following logs messages occur frequently and are not related to short term traffic spikes (such as after a power recovery), you may want to consider increasing the value of the attribute:<br><br>`4721 DHCP ERROR "DHCP has used all xx response packets. A request was dropped and they will continue to be dropped if no responses are available."`<br>`5289 DHCP WARNING "DHCP has used xx of yy response packets. Requests will be dropped if no responses are available."`<br><br>Required. The default is 1000. |
| *max-ping-packets* | set/unset | Controls the number of buffers that the server has available to initiate Ping requests to clients. If you enable the *Ping address before offering it* option at the scope level, packet buffers are used to send and receive ICMP messages. If you enable pinging, you should have enough ping packets allocated to handle the peak load of possible ping requests. The default is 500 ping packets. |
| *defer-lease-extensions* | enable/disable | Controls whether the DHCP server extends leases that are less than half expired. This is a performance tuning attribute that helps minimize the number of disk writes to the lease state database. The default is checked or true. This means that a client renewing a lease less than halfway through can get the remaining part of it only and not be extended. See Deferring Lease Extensions, on page 8. |

| Advanced Parameter | Action | Description |
|---|---|---|
| *last-transaction-time-granularity* | set/unset | The default value of the *last-transaction-time-granularity* attribute has changed from 60 seconds to one week. This new default means that the client-last-transaction-time may not accurately reflect the last time the client communicated with the server.<br><br>If your deployment depended on this attribute being updated whenever the client communicated with the server, you need to explicitly set the *last-transaction-time-granularity* attribute to a value appropriate for the deployment.<br><br>The *last-transaction-time-granularity* attribute is effectively not used when you have disabled defer-lease-extensions. Therefore, if you have disabled defer-lease-extensions, this change in the default value does not impact you.<br><br>When the server is heavily loaded and has run low on request or response buffers, the server temporarily sets the *last-transaction-time-granularity* value to one year to reduce its load. |

| Advanced Parameter | Action | Description |
|---|---|---|
| *discover-queue-limit* | set/unset | Specifies the percentage limit of the request buffers that may be used for DHCPDISCOVER and SOLICIT client requests at any time. Once the configured percentage of the request buffers is exceeded, additional DHCPDISCOVER and SOLICIT client requests are discarded. By restricting the requests buffers that can be used by DHCPDISCOVER/SOLICIT requests, the server assures it has request buffers available to process DHCPREQUEST/REQUEST requests and this can greatly reduce the time needed to get clients online during spikes in activity, such as after a power recovery or CMTS reboot.<br><br>The DRL (Discriminating Rate Limiter) attribute controls the discriminating rate limiter capability. The Discriminating Rate Limiter is enabled by default and assures that the DHCP server prefers completing DHCP transaction over starting too many new ones. In many situations, this should expedite bringing all clients online. If activity summary logging is enabled, the number of DHCPDISCOVER (DHCPv4) and SOLICIT (DHCPv6) packets dropped because of rate limiting is reported as DRL:number.<br><br>The DHCPv4 statistics includes a new queue-limited-discovers-dropped counter and the DHCPv6 statistics includes a new queue-limited-solicits-dropped counter. These counters are used to monitor the packets that are dropped. |

## Local Web UI

**Step 1** From the **Deploy** menu, choose **DHCP Server** under the **DHCP** submenu to open the Manage DHCP Server page.

**Step 2** Select the server from the DHCP Server pane.

**Step 3** Add or modify attributes on the Edit Local DHCP Server page.

**Step 4** Click **Save** after making the changes.

## CLI Commands

Use **dhcp show** and **dhcp get** *attribute* to show the current server parameters, then use **dhcp set** *attribute=value* [*attribute=value ...*], **dhcp unset** *attribute*, **dhcp enable** *attribute*, and **dhcp disable** *attribute* to change them (see the table above).

# Deferring Lease Extensions

Enabling the *defer-lease-extensions* attribute (which is its preset value) allows the DHCP server to optimize response to a sudden flood of DHCP traffic. An example of a network event that could result in such a traffic spike is a power failure at a cable internet service provider (ISP) data center that results in all of its cable modem termination systems (CMTS) rebooting at once. If this happens, the devices attached to the CMTSs produce a flood of DHCP traffic as they quickly come back online.

With the *defer-lease-extensions* attribute enabled, the DHCP server might defer extending the lease expiration time for a client's renewal request, which typically occurs before T1 (usually before halfway through the lease). Instead of giving the client the full configured lease time, the server grants the remaining time on the existing lease. Because the absolute lease expiration time does not change, the server can avoid database updates that result in a significantly higher server throughput. Another benefit is avoiding having to update the failover partner with an extended lease expiration time.

If a client is at or beyond T1 (typically halfway to its expiration), enabling or disabling this attribute has no effect, and the server always tries to extend the lease expiration time. However, failover and other protocol restrictions can prevent the server from extending the lease for the full configured time.

**Note** Deferring lease extensions significantly increases the server performance while remaining in compliance with the DHCP RFC, which stipulates that client binding information is committed to persistent storage when the lease changes.

When deferring lease extensions, it is advisable to leave the policy attribute *allow-lease-time-override* to its default of disabled, or to change it to disabled if it is enabled.

These three specific situations are described from the server point of view:

- **Client retries**—When the server gets behind, it is possible for a client to retransmit requests. The DHCP server does not maintain enough information to recognize these as retransmissions, and processes each to completion, granting a full lease duration again and updating the database. When the server is already behind, doing extra work worsens the situation. To prevent this, the DHCP server does not extend leases that are less than 30 seconds old, regardless of the state of the *defer-lease-extensions* attribute.
- **Client reboots**—The effective renew time for a client lease is really the minimum of the configured renew time and the time between client reboots. In many installations this may mean that clients get fresh leases one (in a typical enterprise) or two (in a typical cable network) times per day, even if the renew time is set for many days. Setting the *defer-lease-extensions* attribute can prevent these early renews from causing database traffic.
- **Artificially short renewal times**—Because there is no way for a DHCP server to proactively contact a DHCP client with regard to a lease, you might configure short lease times on the DHCP server to provide a means of doing network renumbering, address reallocation, or network reconfiguration (for example, a change in DNS server address) in a timely fashion. The goal is to allow you to do this without incurring unacceptable database update overhead.

As a complication, the server also keeps track of the time when it last heard from the client. Known as the last transaction time, sites sometimes use this information as a debugging aid. Maintaining this time robustly requires a write to the database on every client interaction. The *last-transaction-time-granularity* attribute is the one to set. (See the attribute description in Table 1: DHCP Advanced Attributes.) Because it is primarily a debugging aid, the value need not be entirely accurate. Furthermore, because the in-memory copy is always

accurate, you can use **export leases –server** to display the current information, even if the data is not up to date in the database.

# Setting DHCP Forwarding

The Cisco Prime Network Registrar DHCP server supports forwarding DHCP packets to another DHCP server on a per-client basis. For example, you might want to redirect address requests from certain clients, with specific MAC address prefixes, to another DHCP server. This can be useful and important in situations where the server being forwarded to is not one that you manage. This occurs in environments where multiple service providers supply DHCP services for clients on the same virtual LAN.

Enabling DHCP forwarding requires implementing an extension script. The DHCP server intercepts the specified clients and calls its forwarding code, which checks the specified list of forwarded server addresses. It then forwards the requests rather than processing them itself. You attach and detach extensions to and from the DHCP server by using **dhcp attachExtension** and **dhcp detachExtension**.

The DHCP forwarding feature works like this:

1. When DHCP is initialized, the server opens a UDP socket, which it uses to send forwarded packets. To support servers with multiple IP addresses, the socket address pair consists of INADDR_ANY and any port number. This enables clients to use any one of the server IP addresses.
2. When the DHCP server receives a request from a client, it processes these extension point scripts:

   - **post-packet-decode**
   - **pre-client-lookup**
   - **post-client-lookup**

   As the DHCP server processes these scripts, it checks the environment dictionary for this string:

   ```
   cnr-forward-dhcp-request
   ```

3. When it finds that string and it has the value *true* (enabled), the server calls its forwarding code.
4. The forwarding code checks the environment dictionary for a string with this key:

   ```
   cnr-request-forward-address-list
   ```

   It expects a list of comma-separated IP addresses with an optional colon-delimited port number, as in this example:

   ```
   192.168.168.15:1025,192.168.169.20:1027
   ```

   By default, the server forwards to server-port for DHCPv4 and v6-server-port for DHCPv6. It sends a copy of the entire client request to each IP address and port in turn. If any element in the list is invalid, the server stops trying to parse the list.
5. After the forwarding code returns, the server stops processing the request. In the post-client-lookup extension point script, however, this action might create an optional log message with client-entry details.

   The following example of a portion of a TCL extension script tells the DHCP server to forward a request to another server based on the information in the request. You can use such a script if there are multiple device provisioning systems in the same environment. In this case, you would run the extension script on the DHCP server to which routers forward broadcast requests. The script would determine which (if any) other server or servers should handle the request, and tell the original server to forward the request.

   The sample script uses a static mapping of MAC address prefix to send modems from a specific vendor to a specific system:

```
proc postPktDecode {req resp env} {
  set mac [$req get chaddr]
  set addrs ""
  # Very simple, static classifier that forwards all requests from devices
  # with a mac-address vendor-id of 01:0c:10 to the DHCP servers at
  # 10.1.2.3 and 10.2.2.3:
  switch -glob -- $mac {
    01:0c:10* {
      set addrs "10.1.2.3,10.2.2.3"
    }
  }
  # If we decide to forward the packet, the $addrs var will have the IP
  # addresses where to forward the packet:
  if {$addrs != ""} {
    # Tell the DHCP server to forward the packet...
    $env put cnr-forward-dhcp-request true
    # ...and where to forward it:
    $env put cnr-request-forward-address-list $addrs
    # No more processing is required.
    return
  }
}
```

A more flexible script could use a per-client configuration object, such as the Cisco Prime Network Registrar client entry, to indicate which DHCP server should get the request.

**Note** DHCP forwarding is available only for DHCPv4; not for DHCPv6.

# Editing DHCPv6 Server Attributes

You can edit DHCP server attributes related to DHCPv6. These attributes are:

- *v6-client-class-lookup-id*—Expression that determines a client-class based on the DHCPv6 client request and returns a string with either the name of a configured client-class or <none> (if the expression does not wish to provide a client-class). The attribute has no preset value.
- *max-client-leases*—Maximum number of leases a DHCPv6 client can have on a link. Do not use this attribute to limit clients to one lease only. The preset is 50.

## Local Web UI

From the **Deploy** menu, choose **DHCP Server** under the **DHCP** submenu to open the Manage DHCP Server page. Click the **Local DHCP Server** link to open the Edit DHCP Server page, modify the aforementioned DHCPv6 attribute values, then click **Save**.

## CLI Commands

Use **dhcp** to show the aforementioned DHCPv6 server attributes, then modify them by using **dhcp set** *attribute=value* [*attribute=value* ...].

# Using Extensions to Affect DHCP Server Behavior

Cisco Prime Network Registrar provides the ability to alter and customize the operation of the DHCP server through *extensions*, programs that you can write in TCL or C/C++. Extensions interact with the server in two ways: by modifying request or response packets, and through environment variables stored in the environment dictionary (see Using Extension Points for details).

For example, you might have an unusual routing hub that uses BOOTP configuration. This device issues a BOOTP request with an Ethernet hardware type (1) and MAC address in the *chaddr* field. It then sends out another BOOTP request with the same MAC address, but with a hardware type of Token Ring (6). The DHCP server normally distinguishes between a MAC address with hardware type 1 and one with type 6, and considers them to be different devices. In this case, you might want to write an extension that prevents the DHCP server from handing out two different addresses to the same device.

You can solve the problem of the two IP addresses by writing either of these extensions:

- One that causes the DHCP server to drop the Token Ring (6) hardware type packet.
- One that changes the Token Ring packet to an Internet packet and then switches it back again on exit. Although this extension would be more complex, the DHCP client could thereby use either return from the DHCP server.

## Writing Extensions

You can write extensions in TCL or C/C++:

- **TCL**—Makes it a bit easier and quicker to write an extension. If the extension is short, the interpreted nature of TCL does not have a serious effect on performance. When you write an extension in TCL, you are less likely to introduce a bug that can crash the server.
- **C/C++**—Provides the maximum possible performance and flexibility, including communicating with external processes. However, the complexity of the C/C++ API is greater and the possibility of a bug in the extension crashing the server is more likely than with TCL.

You create extensions at specific extension points. Extension points include three types of dictionaries—request, response, and environment. One or more of these dictionaries are available for each of the following extension points:

1. **init-entry**—Extension point that the DHCP server calls when it configures or unconfigures the extension. This occurs when starting, stopping, or reloading the server. This entry point has the same signature as the others for the extension. It is required for DHCPv6 processing. Dictionary: environment only.
2. **pre-packet-decode**—First extension point that the DHCP server encounters when a request arrives, and calls it before decoding the packet. Dictionaries: request and environment.
3. **post-packet-decode**—Rewrites the input packet. Dictionaries: request and environment.
4. **post-class-lookup**—Evaluates the result of a *client-class-lookup-id* operation on the client-class. Dictionaries: request and environment.
5. **pre-client-lookup**—Affects the client being looked up, possibly by preventing the lookup or supplying data that overrides the existing data. Dictionaries: request and environment.
6. **post-client-lookup**—Reviews the operation of the client-class lookup process, such as examining the internal server data structures filled in from the client-class processing. You can also use it to change any data before the DHCP server does additional processing. Dictionaries: request and environment.
7. **generate-lease**—Generates and controls a DHCPv6 address or prefix. Dictionaries: request, response, and environment.

8. **check-lease-acceptable**—Changes the results of the lease acceptability test. Do this only with extreme care. Dictionaries: request, response, and environment.
9. **lease-state-change**—Determines when the lease state changes this only with extreme care. Dictionaries: response and environment.
10. **pre-packet-encode**—Changes the data sent back to the DHCP client in the response, or change the address to which to send the DHCP response. Dictionaries: request, response, and environment.
11. **post-packet-encode**—Allows the server to examine and alter the packet before it sends the packet to the client, or drops the packet. Dictionaries: request, response, and environment.
12. **post-send-packet**—Used after sending a packet for processing that you want to perform outside of the serious time constraints of the DHCP request-response cycle. Dictionaries: request, response, and environment.
13. **environment-destructor**—Allows an extension to clean up any context that it might be holding. Dictionary: environment.

To extend the DHCP server, do the following:

**Step 1** Write the extension in Tcl, C, or C++, and install it in the server extensions directory:

- **Tcl**—/var/nwreg2/local/extensions/dhcp/tcl
- **C or C++**—/var/nwreg2/local/extensions/dhcp/dex

It is best to place these extensions in the appropriate directory for TCL or C/C++ extensions. Then, when configuring the filename, just enter the filename itself, without slash (/).

If you want to place extensions in subdirectories, enter the filename with a path separator.

**Note** Extensions written by you should be added to the /var/nwreg2/local/extensions/dhcp/... area. Extensions shipped with Cisco Prime Network Registrar will be in the /opt/nwreg2/local/extensions/dhcp... area. The server looks for extensions first in the /var/nwreg2/local/extensions/dhcp/... directories and then in the /opt/nwreg2/local/extensions/dhcp/... directories.

**Step 2** Use the List/Add DHCP Extensions page in the web UI (In the Advanced mode, from the **Deploy** menu, choose **Extensions** under the **DHCP** submenu to open the List/Add DHCP Extensions page) or the **extension** command in the CLI to configure the DHCP server to recognize this extension.

**Step 3** Attach the configured extension to one or more DHCP extension points by using **dhcp attachExtension**.

**Step 4** Reload the server.

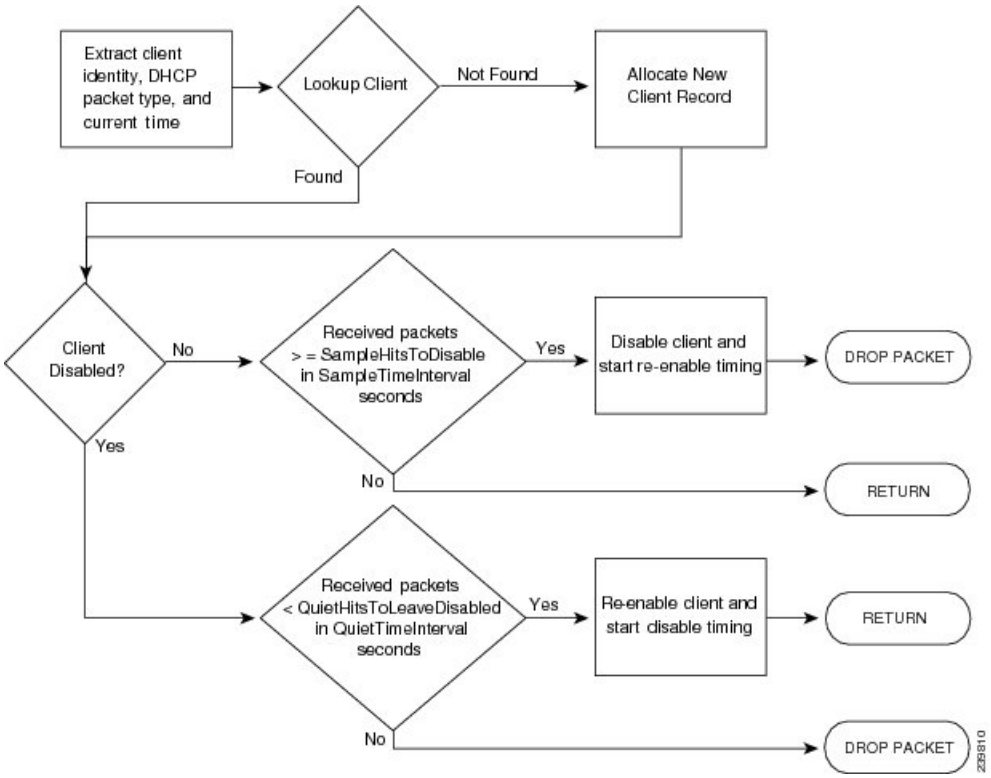
# Preventing Chatty Clients by Using an Extension

One example of an effective use of an extension is to protect against clients flooding the server with unnecessary traffic. You can use the ChattyClientFilter extension to keep the server from having to do much of the work of processing these chatty client packets. If you have large numbers of clients in your network, you might want to consider implementing this extension.

The ChattyClientFilter extension is available in the *install-path*/examples/dhcp/dex directory of the Cisco Prime Network Registrar installation, and compiled and ready to use in *install-path*/extensions/dhcp/dex/dexextension.so or *install-path*/extensions/dhcp/dex/dexextension.dll. The extension monitors client requests, based on the MAC address, and disables the client if it generates more than a certain number of packets in a time interval. Disabling a client means that the server discards packets

from it. However, the server does not ignore the client entirely, because it continues to monitor traffic from it. If the server detects that the client starts to generate fewer than a certain number of packets in a time interval, it reenables the client and begins to allow packets from it again.

***Figure 1: Chatty Client Filter Flow***




The criteria for disabling and reenabling are set through arguments to the ChattyClientFilter extension. By default, the server disables a client when it receives more than 15 packets within 30 seconds; the server reenables the client when it sends fewer than 5 packets within 10 seconds. Note that these defaults are conservative and do not protect against all situations. For example, the server does not disable a client that sends packets every three seconds. Even allowing for a few retransmissions, a client should never need to send more than a half dozen packets in a short interval.

If you suspect chatty clients, review the DHCP server logs to determine incoming rates, then set the arguments described in the table below in the ChattyClientFilter code appropriately.

***Table 2: ChattyClientFilter Arguments***

| ChattyClientFilter Argument | Description |
|---|---|
| **–c** | Ignores the packets when the *drop* attribute of the environment dictionary is set to "true"; default is not to ignore. |

| ChattyClientFilter Argument | Description |
|---|---|
| **–d** *packet-count seconds* | Drops DHCPRELEASE packets if more than the specified count are received in the specified time interval; default disabled.<br>The server keeps dropping DHCPRELEASE packets until the client suspends sending them for the specified interval. (DHCPv4 clients only.)<br>The basic formula is that the time interval should be at least (*packet-count* + 2) * 30 seconds. |
| **–h** *packet-count* | SampleHitsToDisable; default 15 packets. |
| **–i** *seconds* | SampleTimeInterval; default 30 seconds. |
| **–l** *packet-count* | QuietHitsToLeaveDisabled; default 5 packets. |
| **–m** *seconds* | Sets the maximum time a client is disabled, in seconds; default 0 - unlimited. |
| **–n** | NAKs the client if renewing or rebinding; default off. If the client exceeding the SampleHitsToDisable rate does a DHCPREQUEST, the server sends it a DHCPNAK instead of discarding the packet.<br>This can resolve problems with clients (such as cable modems) that cannot renew leases for some reason. Sending the DHCPNAK causes the client to restart its DHCP state machine and send a DHCPDISCOVER.<br>If you use this argument, you must attach the ChattyClientFilter to the **check-lease-acceptable** extension point. (DHCPv4 clients only.) |
| **–q** *seconds* | QuietTimeInterval; default 10 seconds. |
| **–r** *seconds* | StatisticsInterval; default 300 seconds (5 minutes). This argument controls the frequency of periodic logging of the number of clients disabled and reenabled. |
| **–s** | Silently discards dropped packets; default off. |
| **-t** | If specified, the Chatty Client Filter will ignore the “time” in the DUID-LLT for DHCPv6 clients (client-id option (1)). The filter will change the time to zero (00:00:00:00) in the DUID-LLT internally.<br>This can be used in cases where clients using the DUID-LLT are not correctly saving this value and therefore generate multiple client identities.<br>**Note:** This does NOT alter how the server processes these clients. |
| **–w** *port* | Enables web access on specified port. If a positive port number is specified, connections are restricted to the local host. If a negative port number is specified, the absolute value of the port is used and connections are not restricted - this can be a security concern and is not recommended. |
| **–4** | Filters only DHCPv4 packets; default is both. |
| **–6** | Filters only DHCPv6 packets; default is both. |

**Note** The -h, -i, -l, and -q defaults are unlikely to be appropriate to most situations as these were designed to address a single type of misbehaving client. Using a longer interval and packet hit count for normal conditions will produce reasonable results. Values such as -i 120 -h 8 -q 120 -l 8 would allow a client 8 packets over a 120 second period. A normal DHCPDISCOVER/OFFER/REQUEST/ACK is only 2 packets from a client. That is, the proper use of the ChattyClientFilter requires tuning these values for your particular network conditions. Use of the logscan tool which is available from the Cisco Prime Network Registrar download section on the Cisco website can help in analyzing client activity.

Review the comments in the ChattyClientFilter.cpp file for details on setting the arguments and enabling the extension. In most cases, you would attach it to the **post-packet-decode** extension point (along with **check-lease-acceptable** if you use the **–n** argument).

A sample use for the ChattyClientFilter is to drop DHCPRELEASE packets sent from a DHCPv4 client to prevent the lease history database from growing out of bounds, which can be the case with certain router configurations.

This scenario uses the **–d** argument. The setup might be:

```
nrcmd> extension dexChattyClientFilter create dex libdexextension.so
dexChattyClientFilter
init-entry=dexChattyClientFilterInitEntry
init-args="-d 2 120"

nrcmd> dhcp attachextension post-packet-decode dexChattyClientFilter
```

This setup results in the server dropping DHCPRELEASE packets if it receives more than two of these packets from the same client in a 120-second interval, and resuming DHCPRELEASEs processing when the client does not send a DHCPRELEASE for at least 120 seconds.

Cisco Prime Network Registrar supports the mini-web server that can be used to obtain information about the clients being monitored or disabled (traffic being dropped) by the Chatty Client Filter. A typical request might be http://127.0.0.1:<port>/report entered in a web browser.

The web server supports the following requests:

- **status**—Returns a statistics report.
- **report**—Returns a statistics report and a full client report. The client report includes all clients currently being monitored and those that are disabled.
- **disabled-report**—Same as report except only the disabled clients are returned.
- **flush**—Same as report but all clients are REMOVED from the internal monitored and disabled list.
- **csv-client-list**—Returns the client list using CSV format (includes monitored and disabled clients).
- **csv-disabled-client-list**—Same as csv-client-list but only includes clients currently disabled.
- **xml-client-list**—Returns the client list using XML (includes monitored and disabled clients).
- **xml-disabled-client-list**—Returns the disabled client list using XML.

**Note** This web server is a very basic server implementation. It only supports the requests mentioned above.

# Tuning the DHCP Server

Other helpful hints in tuning your DHCP performance include:

- Set the request (*max-dhcp-requests*) and response (*max-dhcp-responses*) buffers for optimal throughput. See Table 1: DHCP Advanced Attributes for details.
- Keep the *defer-lease-extensions* attribute enabled. This reduces writes to the database.
- Set the *last-transaction-time-granularity* attribute to at least 60 seconds, optimally a value greater than half your lease interval.
- Disable the *allow-lease-time-override* attribute for policies offering production leases.
- Minimize your logging and debugging settings. If you require logging, use the *log-settings* attribute for the DHCP server with a controlled number of attributes, as described in the table below.

***Table 3: DHCP Log Settings***

| Log Setting (Numeric Equivalent) | Description |
|---|---|
| default (1) | Gives a low level of logging in several parts of the DHCP server. If you unconfigure the default, even this logging will not appear. |
| activity-summary (20) | Causes a summary message to appear every 1 minute. It is useful when many of the no-xxx log settings are enabled, to give some idea of the activity in the server without imposing the load required for a log message corresponding to each DHCP message. The time period for these messages can be configured with the DHCP server property *activity-summary-interval*. |
| client-criteria-processing (9) | Causes a log message to be output whenever a scope is examined to find an available lease or whenever a scope is examined to determine if a lease is still acceptable for a client who already has one. It can be very useful when configuring or debugging client-class scope criteria processing. It causes moderate amount of information to be logged and should not be left enabled as a matter of course. |
| client-detail (8) | Causes a single line to be logged at the conclusion of every client-class client lookup operation. This line will show the composite of the data found for the client as well as the data that found in the client's client-class. It is useful when setting up a client-class configuration and for debugging problems in client-class processing. |
| dns-update-detail (7) | Causes the server to log a message as it sends each DNS update and as it receives replies to update messages. |
| dropped-waiting-packets (15) | If the value of *max-waiting-packets* is non-zero, packets may be dropped if the queue length for any IP address exceeds the value of *max-waiting-packets*. If *dropped-waiting-packets* is set, the server will log a message whenever it drops a waiting packet from the queue for an IP address. |

| Log Setting (Numeric Equivalent) | Description |
|---|---|
| failover-detail (10) | Causes the server to log a single message for most failover transactions. The information logged is very useful for understanding how failover is operating, and should be included if at all possible when sending requests for support regarding failover issues. |
| incoming-packet-detail (4) | Causes the contents of every DHCP packet received by the DHCP server to be interpreted in a human readable way and printed in the log file. This enables the built-in DHCP packet sniffer for input packets. The log files will fill up (and turn over) very rapidly when this setting is enabled. This setting also causes a significant performance impact on the DHCP server and should not be left enabled as a matter of course. |
| incoming-packets (2) | This setting (on by default) causes a single line message to be logged for every incoming packet. This is especially useful when initially configuring a DHCP server or a BOOTP relay, in that an immediate positive indication exists that the DHCP server is receiving packets. |
| ldap-create-detail (13) | Causes log messages to appear whenever the dhcp server initiates a lease state entry create or delete to LDAP server, receives response and retrieves result or error messages. |
| ldap-query-detail (11) | Causes log messages to appear whenever the DHCP server initiates a query to LDAP server, receives response and retrieves result or error messages. |
| ldap-update-detail (12) | Causes log messages to appear whenever the DHCP server initiates an update lease state to LDAP server, receives response and retrieves result or error messages. |
| leasequery (14) | Causes log messages to appear when leasequery packets are processed without internal errors and result in an ACK or a NAK. |
| minimal-config-info (24) | Reduces the number of configuration messages printed when the server starts or reloads. In particular, it will not log a message for every scope. |
| missing-options (3) | This setting (on by default) causes a message to be logged whenever an option requested by a DHCP client has not been configured in a policy and therefore cannot be supplied by the DHCP server. |
| no-dropped-bootp-packets (18) | Causes the single line message normally logged for every BOOTP packet that is dropped to not appear. |
| no-dropped-dhcp-packets (17) | Causes a single line message normally logged for every DHCP packet that is dropped due to DHCP configuration to not appear. (See *no-invalid-packets* for messages associated with packets dropped because they are invalid.) |
| no-failover-activity (19) | Causes normal activity and some warning messages logged for failover to not appear. Serious error log messages will continue to appear independent of this log-setting. |
| no-failover-conflict (25) | Causes conflicts between failover partners to not be logged. |

| Log Setting (Numeric Equivalent) | Description |
|---|---|
| no-invalid-packets (21) | Causes a single line message normally logged for every DHCP packet that is dropped due to being invalid to not appear. (See *no-dropped-dhcp-packets* for messages associated with packets dropped due to DHCP server configuration.) |
| no-reduce-logging-when- busy (22) | Normally, the DHCP server will reduce logging when it becomes very busy (that is, when it has used over 2/3 of the available receive buffers (itself a configurable value)). It will set no-success-messages, no-dropped-dhcp-packets, no-dropped-bootp-packets, no-failover-activity, no-invalid-packets, and clear everything else except activity-summary. If no-reduce-logging-activity is set, then the server will not do this. It will restore the previous settings when the server becomes unbusy (that is, when it has used only 1/3 of the available receive buffers). |
| no-success-messages (16) | Causes the single line message that is normally logged for every successful outgoing DHCP response packet to not appear. It affects logging only for successful outgoing DHCP response packets. |
| no-timeouts (23) | Causes messages associated with timeout of leases or offers not to appear in the log file. |
| outgoing-packet-detail (5) | Causes the contents of every DHCP packet transmitted by the DHCP server to be interpreted in a human readable way and printed in the log file. This enables the built-in DHCP packet sniffer for output packets. The log files will fill up (and turn over) very rapidly when this setting is enabled. This setting also causes a significant performance impact on the DHCP server and should not be left enabled as a matter of course. |
| unknown-criteria (6) | Causes a single line log message to appear whenever a client entry is found which specifies selection criteria that is not found in any scope appropriate for that client's current network location. |
| v6-lease-detail (27) | Causes the server to log individual messages regarding DHCPv6 leasing activity (in addition to or in place of a single message per client transaction depending on no-success-messages, or client timeout event depending on no-timeouts). |

- Consider setting client caching (see Setting Client Caching Parameters).
- Check the server statistics to aid in monitoring server performance (see the *"Displaying Statistics" section in Cisco Prime Network Registrar 11.1 Administration Guide*).
- Consider setting the scope allocation priority (see Configuring Multiple Scopes Using Allocation Priority).
- If pinging hosts before offering addresses, consider adjusting the ping timeout period (see Pinging Hosts Before Offering Addresses).
- To boost performance, consider limiting the number of selection tags.
- If using Lightweight Directory Access Protocol (LDAP) servers, consider the performance issues described in Configuring Cisco Prime Network Registrar to Use LDAP.

- If using DHCP failover, consider using the load balancing feature (see Setting Load Balancing).

**Tip** Be sure to follow any DHCP server attribute changes with a server reload.

# Listing Related Servers for DHCP - Failover, DNS, LDAP, and TCP Listener Servers

If you have related failover, DNS, LDAP, or TCP Listener servers (see Setting Up Failover Server Pairs), you can access the attributes for these servers.

## Local Web UI

On the Failover Pairs page, click the **Manage Failover Servers** tab and then click the **Related Servers** tab or click the **Related Servers** tab on the Manage DHCP Server page (**Operate** > **Servers** > **Manage Servers**) to open the DHCP Related Server Attributes page. This page shows the communication and failover states the servers are in. The following table describes the attributes on this page. (For this page to appear, you must be assigned the central-cfg-admin role with the dhcp-management subrole.)

***Table 4: Attributes for Related Servers***

| Related Server Attribute | Description |
|---|---|
| *Related Server Type* | Type of related server: DHCP, DNS, or LDAP. |
| *Related Server IP Address* | IP address of the related server. For DHCP failover partners, click this link to open the View Failover Related Server page. |
| *Communications* | State of the communication—None, OK, or Interrupted. |
| *Requests* | Applies to DNS or LDAP related servers only, the number of requests from these servers. |
| *State* | For DHCP failover—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, or Recover-done.<br>For High-Availability (HA) DNS—Send-Update, Probe, or ha-state-unknown. Only the server that is successfully updating can be in Send-Update state. The partner server not sending updates is then always in Probe or unknown state. When the DHCP server comes up if there is no client activity, both DNS servers are often in the unknown state. This changes when the DHCP server tries to do DNS updates. |
| *Partner Role* | For DHCP failover only, the failover role of the partner—Main or Backup. |
| *Partner State* | For DHCP failover only, the partner's state—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, or Recover-done. |

| Related Server Attribute | Description |
|---|---|
| *Update Response Complete* | For DHCP failover only, the percentage of completed update responses, valid only if there are outstanding update responses. |

***Table 5: Attributes for DHCP Related Failover Servers***

| DHCP Related Failover Server Attribute | Description |
|---|---|
| **General attributes** | |
| *failover-pair-name* | The name of the failover pair object used to manage this server. |
| *current-time* | Current time on the server returning this object. |
| *comm-state* | None, OK, or Interrupted. |
| *smoothed-time-delta* | The time difference between the local server and the partner server. If the local server time is ahead of the partner server time, the attribute value is positive. If the local server time is behind the partner server time, the attribute value is negative. If the servers are not communicating, the last known attribute value is recorded. |
| *maximum-client-lead-time* | Current maximum client lead time (MCLT) on this system. |
| *sequence-number* | Sequence number unique across failover objects, if different from the sequence in the lease, the lease is considered “not up to date” independent of the sf-up-to-date lease flag. |
| *load-balancing-backup-pct* | The current failover load balancing backup percentage. If the backup percentage is zero, failover load balancing is not in use (disabled). |
| **Local server information** | |
| *our-ipaddr* | IPv4 address of the interface to this server. |
| *our-ip6address* | IPv6 address of the interface to this server. |
| *role* | Failover role of the server returning this object—None, Main, or Backup. |
| *state* | State of the local server—None, Startup, Normal, Communications- interrupted, Partner-down, Potential-conflict, Recover, Paused, or Recover-done. |
| *start-time-of-state* | Time at which the current failover state began. |

| DHCP Related Failover Server Attribute | Description |
|---|---|
| *start-of-comm-interrupted* | Time at which this partner most recently went into communications- interrupted state. This is valid across reloads, while the start-time-of-state never has a time earlier than the most recent server reload. |
| *est-end-recover-time* | Valid if *update-request-in-progress* is not set to None. If it appears, the time at which the server enters the recover- done state if the update request outstanding is complete. If it does not appear, then the server enters recover-done whenever update-request is completed. |
| *use-other-available* | If false or unset, then this server cannot use other-available leases. If true, then the server can use other-available leases. Valid at all times, but should only be true if in partner-down state. |
| *use-other-available-time* | If, in partner-down state, the *use-other-available* is false or unset, the time when *use-other-available* will go to true. |
| *safe-period-remaining* | Duration in seconds remaining in safe-period. If not set to 0, then this server is currently running down a safe period with respect to its partner. |
| *load-balancing-local-hba* | The current hash bucket assignment of the local server, usually shown as a range of the hash bucket numbers. (See RFC 3074.) |
| *request-buffers-in-use* | The number of failover request buffers the DHCP server is using at the time the statistics are calculated. |
| *decaying-max-request-buffers-in-use* | The maximum number of failover request buffers that have recently been in use. |
| *request-buffers-allocated* | The number of request buffers that the server has allocated to support the failover capability. |
| *connection-start-time* | The time at which the most recent connection started. This value is set whenever a connection is started, and it not cleared when a connection ended. |
| *connection-end-time* | The time at which the most recent connection ended. This value is set whenever a connection is ended, and it not cleared when a new connection starts. |
| **Partner server information** | |
| *ipaddr* | IP address of the partner server. |
| *ip6address* | IPv6 address of the partner server. |

| DHCP Related Failover Server Attribute | Description |
|---|---|
| *partner-role* | Failover role of the partner of the server returning this object—None, Main, or Backup. |
| *partner-state* | Last known state which the partner end of the failover relationship is in—None, Startup, Normal, Communications-interrupted, Partner-down, Potential-conflict, Recover, Paused, or Recover-done. |
| *start-time-of-partner-state* | Time at which the partner current failover state began. |
| *est-partner-end-recover- time* | If the *partner-state* is Recover, an estimated prediction of when the partner will time out its MCLT and finish being in recover state. |
| *last-comm-ok-time* | Time at which this server last found communications to be OK. |
| *load-balancing-partner- hba* | The current hash bucket assignment of the partner server, usually shown as a range of the hash bucket numbers. (See RFC 3074.) |
| *partner-vendor-major- version* | The vendor ID major version from the partner server. |
| *partner-vendor-minor- version* | The vendor ID minor version from the partner server. |
| **Update requests sent to partner** | |
| *update-request- outstanding* | If None or unset, then the server does not have an update request queued for its partner. If not set to None, then it does have an update request queued for its failover partner. Valid values are None, Update, and Update-all. |
| *update-request-start-time* | Time at which any *update-request-outstanding* request was started. |
| *update-request-done-time* | Time at which the last of any update request completed. |
| *v6-update-response-in-progress* | The type and origin of the response. |
| *v6-update-response-percent-complete* | The percent complete of the current IPv6 update response. |
| *v6-update-response-start-time* | The time that the IPv6 update response mentioned in v6-update-response-in-progress was started. |
| *v6-update-response-done-time* | The time that the most recent IPv6 update response sent an update done to the partner server. |
| **Update requests processed for partner** | |

| DHCP Related Failover Server Attribute | Description |
|---|---|
| *update-response-in- progress* | If this server is processing an update response, gives information about the type and origin of the response. |
| *update-response-percent- complete* | If *update-response-outstanding* appears, the percent complete of the current update response. |
| *update-response-start- time* | Time that the update response mentioned in *update-response-in-progress* was started. |
| *update-response-done- time* | Time that the most recent update response sent an update done to the partner server. |
| **Load Balancing Counters** | |
| *load-balancing-processed- requests* | The number of server processed requests, both IPv4 and IPv6, subject to load balancing. This counter includes only the requests made after the latest transition of server to NORMAL state. |
| *load-balancing-dropped- requests* | The number of server dropped requests, both IPv4 and IPv6, subject to load balancing. This counter includes only the requests made after the latest transition of server to NORMAL state. |
| *load-balancing-processed- total* | The number of server processed requests, both IPv4 and IPv6, subject to load balancing. This counter includes the requests since this server was last started or reloaded. |
| *load-balancing-dropped- total* | The number of server dropped requests, both IPv4 and IPv6, subject to load balancing. This counter includes the requests since this server was last started or reloaded. |
| **Binding Update or Ack Counters (this connection)** | |
| *binding-updates-sent* | The number of binding update (BNDUPD) messages sent to the failover partner. |
| *binding-acks-received* | The number of binding acknowledgement (BNDACK) messages received from the failover partner. |
| *binding-updates-received* | The number of binding update (BNDUPD) messages received from the failover partner. |
| *binding-acks-sent* | The number of binding acknowledgement (BNDACK) messages sent to the failover partner. |
| *v6-binding-updates-sent* | The number of IPv6 binding updates (BNDUPD6) messages received from the failover partner since the start of the most recently established connection. |

| DHCP Related Failover Server Attribute | Description |
|---|---|
| *v6-binding-acks-received* | The number of IPv6 binding acknowledgements (BNDACK6) messages received from the failover partner since the start of the most recently established connection. |
| *v6-binding-updates-received* | The number of IPv6 binding updates (BNDUPD6) messages received from the failover partner since the start of the most recently established connection. |
| *v6-binding-acks-sent* | The number of IPv6 binding acknowledgements (BNDACK6) messages sent to the failover partner since the start of the most recently established connection. |
| **Binding Update/Ack Counters Totals** | |
| *binding-updates-sent-total* | The number of IPv4 binding updates (BNDUPD) messages sent to the failover partner since the most recent statistics reset. |
| *binding-acks-received-total* | The number of IPv4 binding acknowledgements (BNDACK) messages received from the failover partner since the most recent statistics reset. |
| *binding-updates-received-total* | The number of IPv4 binding updates (BNDUPD) messages received from the failover partner since the most recent statistics reset. |
| *binding-acks-sent-total* | The number of IPv4 binding acknowledgements (BNDACK) messages sent to the failover partner since the most recent statistics reset. |
| *v6-binding-updates-sent-total* | The number of IPv6 binding updates (BNDUPD6) messages sent to the failover partner since the most recent statistics reset. |
| *v6-binding-acks-received-total* | The number of IPv6 binding acknowledgements (BNDACK6) messages received from the failover partner since the most recent statistics reset. |
| *v6-binding-updates-received-total* | The number of IPv6 binding updates (BNDUPD6) messages received from the failover partner since the most recent statistics reset. |
| *v6-binding-acks-sent-total* | The number of IPv6 binding acknowledgements (BNDACK6) messages sent to the failover partner since the most recent statistics reset. |
| **Flow Control Counters (this connection)** | |
| *current-binding-updates-in-flight* | The current number of binding updates (both IPv4 and IPv6) that are currently in-flight (sent). |

| DHCP Related Failover Server Attribute | Description |
|---|---|
| *current-binding-updates-queued* | The current number of binding updates (both IPv4 and IPv6) that are queued at present.<br>This number should typically be small (as a percentage of total leases), but can grow large if there are many leases needing updates (such as if the partners were integrated after an outage of one of them) or the partner is slow to process updates. This number will not exceed the total number of leases.<br>It is a cause for concern if the number is greater than 1000 or 10% of the leases (whichever is larger), and even more so if it is continuing to grow. Typically this means that the failover partner is experiencing high latency in processing requests (usually disk latency is the primary issue). |
| *maximum-binding-updates-in-flight* | The maximum number of binding updates (both IPv4 and IPv6) that were in-flight (sent) at one time. |
| *maximum-binding-updates-queued* | The maximum number of binding updates (both IPv4 and IPv6) that were queued at one time. |
| *last-binding-update-sent-time* | The time the last binding update (either IPv4 or IPv6) was sent. |
| *last-binding-ack-received-time* | The time the last IPv4 or IPv6 binding acknowledgement (whether NAKed or not) was received. |
| *last-binding-update-received-time* | The time the last binding update (either IPv4 or IPv6) was received. |
| *last-binding-ack-sent-time* | The time the last IPv4 or IPv6 binding acknowledgement (whether NAKed or not) was sent. |

***Table 6: Attributes for DNS Related Failover Servers***

| DNS Related Server Attribute | Description |
|---|---|
| **General attributes** | |
| *current-time* | Current time on the server returning this object. |
| *ipaddr* | IP address |

| DNS Related Server Attribute | Description |
|---|---|
| *comm-state* | There are three possible values: None, OK, or Interrupted.<br>Status of communication between the DHCP and remote server. An 'OK' indicates DHCP server succeeded in communicating to the remote server. An 'Interrupted' indicates DHCP server was unsuccessful in communicating to the remote server. |
| *dns-server-state* | There are two possible values: PROBE or SEND-UPDATE.<br>PROBE means that the DHCP server has either not yet tried to communicate to this server, or it was determined to be down and hence it is actively being probed (this means that the DHCP server only sends one update request to it).<br>SEND-UPDATE means that the server appears to be communicating and that the DHCP server can send it many requests. |
| *probe-polling-event-id* | Zero. |
| *requests* | Number of requests currently outstanding with the remote server. |
| **HA DNS Configuration information** | |
| *ha-dns-role* | Role played by this DNS server. The value can be STANDALONE-DNS, HA-MAIN, or HA-BACKUP.<br>A DNS server can be a standalone DNS, or HA-Main or HA-Backup if HA-DNS is in use. |
| *dns-timeout* | Number of milliseconds that the DHCP server will wait for a response from the DNS server for a dynamic dns update, before retrying dynamic dns update. |
| *max-dns-retries* | Number of times that the DHCP server will try to send dynamic updates to a DNS server. |
| *ha-dns-failover-timeout* | Maximum time period, in seconds, the DHCP server will wait for a reply from a DNS server, before the DHCP will failover to use next DNS Server to perform the dynamic-update. Default value is 30 seconds. |
| *ha-dns-probe-timeout* | If cnr-ha-dns is enabled, DHCP server will use this timer to co-oridnate and reduce latency in failing over between HA-DNS servers, when HA-DNS servers are in COMMUNIATION-INTERRUPTED state or SYNCHRONIZING. Default value is 3 seconds. |

| DNS Related Server Attribute | Description |
|---|---|
| *ha-dns-probe-retry* | If cnr-ha-dns is enabled, DHCP server will use this retry count and ha-dns-probe-timeout to co-oridnate and reduce latency in failing over between HA-DNS servers, when HA-DNS servers are in COMMUNIATION-INTERRUPTED state or SYNCHRONIZING. Default value is 1 retry attempt. |

***Table 7: Attributes for TCP Listener and Connection Related Servers***

| TCP Listener and Connection Related Server Attribute | Description |
|---|---|
| **TCP Listener Related Server Attributes** | |
| *ipaddr* | Address to which the listener is bound. This may be 0.0.0.0. |
| *comm-state* | Status of communication. This will always be none. |
| *ip6address* | IPv6 address to which the listener is bound. This may be 0::0. |
| *name* | Name of the service. |
| *port* | Port number to which the listener is bound. Incoming connections to this port will be processed. |
| *total-connections* | Total number of incoming connections. |
| *current-connections* | Number of currently active connections. |
| *rejected-connections* | Total number of incoming connections that were rejected, such as the maximum number of active connections were exceeded. |
| **TCP Connection Related Server Attributes** | |
| *ipaddr* | Address of the remote end of the connection. |
| *comm-state* | Status of communication. This will always be ok. |
| *ip6address* | IPv6 address of the remote end of the connection. |
| *name* | Name of the service on which this connection was accepted. |
| *port* | Port number for the remote end of the connection. |
| *total-requests* | Total number of request messages received. |
| *current-requests* | Number of active requests. |
| *current-state* | Current state of the connection. |
| *total-replies* | Total number of reply messages sent. |
| *start-time* | Time when the connection was established. |

| TCP Listener and Connection Related Server Attribute | Description |
|---|---|
| *last-receive-time* | Time of the last byte received. |
| *last-send-time* | Time of the last byte sent. |
| *total-bytes-received* | Total number of bytes received over the connection. |
| *total-bytes-sent* | Total number of bytes sent over the connection. |
| *our-ipaddr* | Address of the local end of the connection. |
| *our-ip6address* | IPv6 address of the local end of the connection. |
| *our-port* | Port number for the local end of the connection. |

Other controls are available on these pages:

- To refresh the data on the Related Server tab, click **Refresh Data**.
- On the Related Server tab, if the partner is in the Communications-interrupted failover state, you can click **Set Partner Down** in association with an input field for the partner-down date setting. This setting is initialized to the value of the *start-of-communications- interrupted* attribute. (In Normal web UI mode, you cannot set this date to be an earlier value than the initialized date. In Expert web UI mode, you can set this value to any date.) After clicking **Set Partner Down**, you return to the List Related Servers for DHCP Server page to view the result of the partner-down action. Never set both partners to Partner Down mode.
- To return from the List Related Servers for DHCP Server page or View Failover Related Server page, click **Return**.

## CLI Commands

To list the related servers for a DHCP server in a brief table form, with a subset of the values, use **dhcp getRelatedServers**. To report the full details (normal object form display, not table), use **dhcp getRelatedServers full**.

# Configuring Virtual Private Networks

This section describes how to configure the Cisco Prime Network Registrar DHCP server to support virtual private networks (VPNs).

Configuring VPNs involves an adjustment to the usual DHCP host IP address designation. VPNs use private address spaces that might not be unique across the Internet. Because of this, Cisco Prime Network Registrar supports IP addresses that are distinguished by a VPN identifier. Relay agents on routers must support this capability as well. The VPN identifier selects the VPN to which the client belongs. VPN for DHCP is currently only supported by Cisco IOS software, the newest versions of which can include VPN IDs in the relayed DHCP messages.

# Related Topics

Configuring Virtual Private Networks Using DHCP, on page 29

VPN and Subnet Allocation Tuning Parameters, on page 35

# Configuring Virtual Private Networks Using DHCP

VPNs that you create provide a filtering mechanism for:

- Viewing the unified address space (see Viewing Address Space)
- Listing address blocks (see Adding Address Blocks)
- Listing subnets (see Address Blocks and Subnets)
- Querying DHCP utilization (see Querying Utilization History Data)
- Querying lease history (see Running IP Lease Histories)

If you do not configure a VPN, Cisco Prime Network Registrar uses the global VPN of 0 on each scope.

To configure a VPN whereby a client can request IP addresses from a DHCP server using a relay agent, you must define the VPN and associate a scope with it. Specifically:

1. Ensure that the relay agents that handle DHCP VPN traffic are configured with a version of Cisco IOS software that supports the *vpn-id* suboption of the *relay-agent-info* option (82) in DHCP.
2. Coordinate with the Cisco IOS relay agent administrator that the VPN is identified either by a VPN ID or a VPN Routing and Forwarding instance (VRF) name.
3. Create a scope for the VPN.

## Typical Virtual Private Networks

Figure 4 shows a typical VPN scenario with DHCP client 1 as part of VPN blue and DHCP client 2 in VPN red. For example, both DHCP client 1 in VPN blue and client 2 in VPN red have the same private network address: 192.168.1.0/24. The DHCP relay agent has gateway addresses that are in the two VPNs as well as a global one (172.27.180.232). There are two failover DHCP servers, both of which know the relay agent through its external gateway address.

Here is the processing that takes place for the server to issue a VPN-supported address to a client:

1. DHCP client 1 broadcasts a DHCPDISCOVER packet, including its MAC address, hostname, and any requested DHCP options.
2. DHCP relay agent at address 192.168.1.1 picks up the broadcast packet. It adds a *relay-agent-info* option (82) to the packet and includes the *subnet-selection* suboption that identifies 192.168.1.0 as the subnet. The packet also includes the *vpn-id* suboption that identifies the VPN as *blue*. Because the DHCP server cannot communicate directly with the requesting client, the *server-id-override* suboption contains the address of the relay agent as known by the client (192.168.1.1). The relay agent also includes in the packet its external gateway address (*giaddr*), 172.27.180.232.
3. The relay agent unicasts the DHCPDISCOVER packet to the configured DHCP server on its subnet.
4. DHCP server 1 receives the packet and uses the *vpn-id* and *subnet-selection* suboptions to allocate an IP address from the proper VPN address space. It finds the available address 192.168.1.37 in the subnet and VPN, and places it in the *yiaddr* field of the packet (the address offered to the client).
5. The server unicasts a DHCPOFFER packet to the relay agent that is identified by the *giaddr* value.
6. The relay agent removes the *relay-agent-info* option and sends the packet to DHCP client 1.
7. DHCP client 1 broadcasts a DHCPREQUEST message requesting the same IP address that it was offered. The relay agent receives this broadcast message.

8. The relay agent forwards the DHCPREQUEST packet to DHCP server 1, which replies with a unicast DHCPACK packet to the client.
9. For a lease renewal, the client unicasts a DHCPRENEW packet to the IP address found in the *dhcp-server-identifier* option of the DHCPACK message. This is 192.168.1.1, the address of the relay agent. The relay agent unicasts the packet to the DHCP server. The server does its normal renewal processing, without necessarily knowing whether it was the server that gave out the original address in the first place. The server replies in a unicast DHCPACK packet. The relay agent then forwards the DHCPACK packet to the client IP address identified by the *ciaddr* field value.

   If the *server-id-override* suboption of the *relay-agent-info* option (82) exists, the DHCP server uses its value to compare to that of the *dhcp-server-identifier* option in the reply packet. Any packet that the DHCP client unicasts then goes directly to the relay agent and not to the server (which may, in fact, be inaccessible from the client). Both partners in a failover environment can renew a lease if the packet includes the *server-id-override* suboption.

## Creating and Editing Virtual Private Networks

To set up the VPN and its index:

**Step 1** Coordinate with the Cisco IOS relay agent administrator that the VPNs are configured either by VPN ID or VRF name on the relay agent. This will determine how to identify the VPN in Cisco Prime Network Registrar.

**Step 2** Create a VPN to allow provisioning DHCP clients onto the VPN that is configured in the IOS switch or router.

**Step 3** Enter a VPN index, which can be any unique text string except the reserved words **all** or **global**. Its associated ID must also be unique.

To add an index at the:

- **Local cluster (Advanced)**—From the **Design** menu, choose **VPNs** under the **DHCP Settings** submenu to open the List/Add VPNs page. Give the VPN a numerical key identifier and a unique name in the cluster.
- **Regional cluster**—Add the local cluster containing the VPN (from the **Operate** menu, choose **Manage Clusters** under the **Servers** submenu). Then choose **VPNs** from the **Design** menu. This opens the List/Add VPNs page. You can create the VPN on this page or pull the VPN from the local clusters:
  - If creating the VPN, give it a numerical key identifier and a unique name.
  - If pulling the VPN from the local clusters, click the **Pull Data** icon in the VPNs pane on the List/Add VPNs page, then pull a specific VPN or all the VPNs from the selected cluster.

  You can also push VPNs to the clusters by clicking the **Push** or **Push All** icon in the List/Add VPNs page. Then choose the synchronization mode and the clusters to which to push the VPNs on the Push VPN Data to Local Clusters page.
- **In the CLI** —Use **vpn** *name* **create** *key*. For example:

```
nrcmd> vpn blue create 99
```

**Step 4** Specify the appropriate VPN identifier, either by VPN ID or VRF name. It is rarely both.

- If you use a VPN ID, set the *vpn-id* attribute value for the VPN. The value is usually in hexadecimal, in the form *oui***:***index*, per IETF RFC 2685. It consists of a three-octet VPN Organizationally Unique Identifier (OUI) that corresponds to the VPN owner or ISP, followed by a colon. It is then followed by a four-octet index number of the VPN itself. Add the VPN ID value to the List/Add VPNs page. In the CLI, set the *vpn-id* attribute. For example:

```
nrcmd> vpn blue set vpn-id=a1:3f6c
```

- If you use a VPN Routing and Forwarding (VRF) instance name, set the *vrf-name* attribute value for the VPN. Cisco routers frequently use VRF names. Add the VRF Name value to the List/Add VPNs page. In the CLI, set the *vrf-name* attribute. For example:

  ```
  nrcmd> vpn blue set vrf-name=framus
  ```

**Step 5** Add a description for the VPN (optional).

**Step 6** Click **Add VPN**. You can edit the VPN to change the values on the Edit VPN page.

**Step 7** Create a scope for the VPN.

You must keep the VPN name and scope name as similar as possible for identification purposes.

**a.** In the web UI, from the **Design** menu, choose **Scopes** under the **DHCPv4** submenu to open the List/Add DHCP Scopes page.

**b.** Choose the VPN from the VPN submenu under the **Settings** drop-down list at the top of the web UI. You cannot change the VPN once you set it at the time of creation of the scope.

In the CLI, identify to which VPN the scope belongs in one of three ways:

- Its VPN name, through the *vpn* attribute (which applies the VPN ID to the scope).
- The VPN ID itself, through the *vpn-id* attribute.
- The current session VPN name, by omitting the VPN or its ID on the command line.

You set the default VPN for the current session using *session set current-vpn*. You can then set the usual address range and necessary option properties for the scope. For example:

```
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0 vpn=blue
```

Or

```
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0 vpn-id=99
```

Or

```
nrcmd> session set current-vpn=blue
nrcmd> scope blue-1921681 create 192.168.1.0 255.255.255.0
```

Then

```
nrcmd> scope blue-1921681 addRange 192.168.1.101 192.168.1.200
nrcmd> scope-policy blue-1921681 setOption routers 192.168.1.1
```

If you are in the staged dhcp edit mode, reload the DHCP server after you create all the VPNs and scopes.

## VPN Usage

The VPN name is used to qualify many DHCP objects in Cisco Prime Network Registrar, such as IP addresses (leases), scopes, and subnets. For example, lease names can have this syntax:

*vpn*/*ipaddress*

For example, red/192.168.40.0

A VPN can be any unique text string except the reserved words **global** and **all**. You can use **global** and **all** when you lease data. The **global** VPN maps to the [none] VPN; the **all** VPN maps to both the specific VPN and the [none] VPN.

In the CLI, if you omit the VPN or its ID in defining an object, the VPN defaults to the value set by **session set current-vpn**. In the web UI, if the current VPN is not defined, it defaults to the [none] VPN, which includes all addresses outside of any defined VPNs.

These objects have associated VPN properties:

- **Address blocks**—Define the VPN for an address block. Choose **Address Blocks** from the **Design > DHCPv4** menu to open the List/Add DHCP Address Blocks page (available in Advanced mode). Choose the VPN from the VPN submenu under the **Settings** drop-down list at the top of the web UI. In the CLI, use the **dhcp-address-block** creation and attribute setting commands. For example:

```
nrcmd> dhcp-address-block red create 192.168.50.0/24

nrcmd> dhcp-address-block red set vpn=blue

nrcmd> dhcp-address-block red set vpn-id=99
```

**Note** Before creating the objects, set the vpn-id value to the VPN in which the dhcp-address-block has to be created. Do not assume that vpn-id is always the current VPN.

- **Clients and client-classes**—In some cases it is best to provision a VPN inside of Cisco Prime Network Registrar instead of externally, where it might have to be configured for every Cisco IOS device. To support this capability, you can specify a VPN for a client or client-class. Two attributes are provided:
  - *default-vpn* —VPN that the packet gets if it does not already have a *vpn-id* or *vrf-name* value in the incoming packet. You can use the attribute with clients and client-classes.
  - *override-vpn* —VPN the packet gets no matter what is provided for a *vpn-id* or *vrf-name* value in the incoming packet. You can use the attribute with clients and client-classes. Note that if you specify an override VPN on the client-class, and a default VPN for the client, the override VPN on the client-class takes precedence over the default VPN on the client.

  At the local cluster—Choose **Clients** or **Client Classes** (available in Advanced mode) from the **Design > DHCP Settings** menu. Create or edit a client-class or client and enter the *default-vpn* and *override-vpn* attribute values.

  At the regional cluster—Choose **Client Classes** (available in Advanced mode) from the **Design > DHCP Settings** menu. Create or pull, and then edit a client-class to enter the *default-vpn* and *override-vpn* attribute values.

  In the CLI—Use the **client-class** creation and attribute setting commands. For example:

```
nrcmd> client 1,6,00:d0:ba:d3:bd:3b set default-vpn=blue

nrcmd> client-class CableModem set override-vpn=blue
```

  In a cable modem deployment, for example, you can use the *override-vpn* attribute to provision the cable modems. The client-class would determine the scope for the cable modem, and the scope would determine the VPN for the uBR. User traffic through the cable modem would then have the *vpn-id* suboption set and use the specific VPN. The *override-vpn* value also overrides any *default-vpn* set for the client.

- **Leases**—List leases, show a lease, or get lease attributes.

  In the CLI—To import leases, use **import leases** *filename*. Each lease entry in the file can include the VPN at the end of the line. If it is missing, Cisco Prime Network Registrar assigns the [none] VPN. (See also Importing and Exporting Lease Data.)

  ```
  nrcmd> import leases leaseimport.txt
  ```

  To export the address or lease data to include the VPN, use **export leases** with the **–vpn** option. The VPN value can be the reserved word **global** or **all**:

  - **Global**—Any addresses outside the defined VPNs (the [none] VPN).
  - **All**—All VPNs, including the [none] VPN.

  If you omit the VPN, the export uses the current VPN as set by **session set current-vpn**. If the current VPN is not set, the server uses the [none] VPN.

  ```
  nrcmd> export addresses file=addrexport.txt vpn=red
  ```

  ```
  nrcmd> export leases -server -vpn red leaseexport.txt
  ```

- **Scopes**—Scopes can include the VPN name or its ID, as described in Configuring Virtual Private Networks Using DHCP, on page 29.

  **Note** You cannot change the VPN once you set it at the time of creation of the scope.

- **Subnets**—Listing subnets, showing a subnet, or getting the *vpn* or *vpn-id* attribute for a subnet shows the VPN. See Configuring DHCP Subnet Allocation, on page 34.

- **DHCP server**—If the *vpn-communication* attribute is enabled (which it is by default), the DHCP server can communicate with DHCP clients that are on a different VPN from that of the DHCP server by using an enhanced DHCP relay agent capability. This capability is indicated by the *server-id-override* suboption in the relay agent information option (82).

  **Note** The DHCP server does not try to ping the clients residing in VPNs.

# Configuring Subnet Allocation

This section describes how to configure the Cisco Prime Network Registrar DHCP server to support subnet allocation for on-demand address pools.

Subnet allocation is a way of leasing subnets to clients (usually routers or edge devices) so that they can, in turn, provide DHCP services. This can occur along with or instead of managing individual client addresses. Subnet allocation can vastly improve IP address provisioning, aggregation, characterization, and distribution by relying on the DHCP infrastructure to dynamically manage subnets. Subnet allocation through DHCP is currently only supported by Cisco IOS software, the newest versions of which incorporate the on-demand address pools feature.

**Note** DHCP failover does not include DHCPv4 subnet allocation.

# Configuring DHCP Subnet Allocation

The following section provides an example of setting up subnet allocation using the DHCP server. Figure 5 shows a sample subnet allocation configuration with subnets assigned to provisioning devices, along with the conventional DHCP client/server configuration.

Before allocating subnets, the DHCP server first determines what VPN the client is on, in the following order:

1. The server looks for incoming VPN options and uses the value for the VPN.
2. If no VPN options are found, the server uses the relay agent suboption value, then combines the VPN with the subnet address to form the unique identifier.
3. If no relay agent suboption is found, the server looks for client-class information (selection tags).

To configure DHCP subnet allocation:

**Step 1** Create a DHCP address block for a subnet, set the initial subnet mask and its increment, and set other subnet allocation request attributes. Also, associate a policy or define an embedded policy.

- If you use VPNs, you can specify a *vpn* or *vpn-id* attribute (see Configuring Virtual Private Networks Using DHCP, on page 29).
- The server uses the presence of the *subnet-alloc* DHCP option (220) in the request packet to determine that the packet is a subnet allocation request. You can configure the server to use the *subnet-name* suboption (3) as a selection tag if you set the *addr-blocks-use-selection-tags* attribute for the server or VPN.
- You can optionally set a default selection tag by setting the *addr-blocks-default-selection-tags* attribute for the DHCP server or VPN object. This identifies one or more subnets from which to allocate the addresses. If the relay agent sends a VPN string (via a VPN option or relay agent suboption), associated with a subnet, any address block with that string as one of its *addr-blocks-default-selection-tags* values uses that subnet.
- The default behavior on the server and for VPNs is that the DHCP server tries to allocate subnets to clients using address blocks that the clients already used. Disabling the *addr-blocks-use-client-affinity* attribute causes the server to supply subnets from any suitable address block, based on other selection data in the clients' messages.
- If you want to support configurations of multiple address blocks on a single LAN segment (analogous to using primary and secondary scopes), add a *segment-name* attribute string value to the DHCP address block. When the relay agent sends a single subnet selection address, it selects address blocks tagged with that *segment-name* string value. However, you must also explicitly enable the LAN segment capability (*addr-blocks-use-lan-segments*) at the server or VPN level.
- Instead of associating a policy, you can set properties for the address block embedded policy. As in embedded policies for clients, client-classes, and scopes, you can enable, disable, set, unset, get, and show attributes for an address block policy. You can also set, unset, get, and list any DHCP options for it, as well as set, unset, and list vendor options. Note that deleting an address block embedded policy unsets all the embedded policy properties.

**Step 2** Note that the server allocates subnets based on the relay agent request. If not requested, the default subnet size is a 28-bit address mask. You can change this default, if necessary, by setting the *default-subnet-size* attribute for the DHCP address block.

For example:

```
nrcmd> dhcp-address-block red set default-subnet-size=25
```

**Step 3** You can control any of the subnets the DHCP server creates from the address blocks. Identify the subnet in the form *vpn-name*/*netipaddress*/*mask*, with the *vpn-name* optional. Subnet control includes activating and deactivating the subnet as you would a lease. Likewise, you can force a subnet to be available, with the condition that before you do so, that you check that the clients assigned the subnet are no longer using it. First, show any subnets created.

**Step 4** Reload the DHCP server.

## VPN and Subnet Allocation Tuning Parameters

Consider these tuning parameters for VPNs and on-demand address pools.

- **Keep orphaned leases that have nonexistent VPNs**—Cisco Prime Network Registrar usually maintains leases that do not have an associated VPN in the Cisco Prime Network Registrar state database. You can change this by enabling the DHCP attribute *delete-orphaned-leases*. The server maintains a lease state database that associates clients with leases. If a scope modification renders the existing leases invalid, the lease database then has orphaned lease entries. These are typically not removed even after the lease expires, because the server tries to use this data in the future to reassociate a client with a lease. One downside to this is that the lease database may consume excessive disk space. When you enable the *delete-orphaned-leases* attribute, such lease database entries are removed during the next server reload. However, be cautious when enabling this attribute, because rendering leases invalid can result in clients using leases that the server believes to be free. This can compromise network stability.
- **Keep orphaned subnets that have nonexistent VPNs or address blocks**—This is the default behavior, although you can change it by enabling the DHCP attribute *dhcp enable delete-orphaned-subnets*. As the DHCP server starts up, it reads its database of subnets and tries to locate the parent VPN and address block of each subnet. With the attribute enabled, if a subnet refers to a VPN that is no longer configured in the server, or if the server cannot locate a parent address block that contains the subnet, the server permanently deletes the subnet from the state database.
- **Keep the VPN communication open**—This is the default behavior, although you can change it by disabling the DHCP attribute *vpn-communication*. The server can communicate with clients that reside on a different VPN from that of the server by using an enhanced DHCP relay agent capability. This is signaled by the appearance of the *vpn-id* suboption of the *relay-agent-info* option (82). You can disable the *vpn-communication* attribute if the server is not expected to communicate with clients on a different VPN than the server. The motivation is typically to enhance network security by preventing unauthorized DHCP client access.

# Configuring BOOTP

BOOTP (the BOOTstrap Protocol) was originally created for loading diskless computers. It was later used to allow a host to obtain all the required TCP/IP information to use the Internet. Using BOOTP, a host can broadcast a request on the network and get information required from a BOOTP server. The BOOTP server is a computer that listens for incoming BOOTP requests and generates responses from a configuration database for the BOOTP clients on that network. BOOTP differs from DHCP in that it has no concept of lease or lease expiration. All IP addresses that a BOOTP server allocates are permanent.

You can configure Cisco Prime Network Registrar to act like a BOOTP server. In addition, although BOOTP normally requires static address assignments, you can choose to either reserve IP addresses (and, therefore, use static assignments) or have IP addresses dynamically allocated for BOOTP clients.

# About BOOTP

When you configure the DHCP server to return a BOOTP packet, be aware that BOOTP requires information in the DHCP packet in fields other than the option space. BOOTP devices often need information in the boot file (*file*), server IP address (*siaddr*), and server hostname (*sname*) fields of the DHCP packet (see RFC 2131).

Every Cisco Prime Network Registrar DHCP policy has attributes with which you can configure the information you want returned directly in the *file*, *siaddr*, or *sname* fields. The Cisco Prime Network Registrar DHCP server also supports a configuration parameter with which you can configure the policy options and determine which of the *file*, *sname*, or *siaddr* values you want returned to the BOOTP device.

Cisco Prime Network Registrar supports an analogous configuration parameter with which you can configure the options and *file*, *sname*, or *siaddr* values you want returned to the DHCP client. This is in addition to any options requested by the DHCP clients in the *dhcp-parameter-request* option in the DHCP request. Thus, you can configure both the BOOTP and DHCP response packets appropriately for your devices.

**Step 1** Decide which values you want for the BOOTP attributes:

- *file*—Name of the boot file
- *siaddr*—Server IP address
- *sname*—Optional server hostname

**Step 2** Decide the list of options and their values that you want returned to the BOOTP client.

**Step 3** Set these values in the policy you want associated with the BOOTP request:

- Attributes (*packet-siaddr*, *packet-file-name*, *packet-server-name*) to send to the BOOTP client.
- Option values, such as the server addresses and domain name to return to the BOOTP client.
- List of fields and options you want returned to the BOOTP client.

**Step 4** Enable the associated scope or scopes for BOOTP processing.

**Step 5** Enable dynamic BOOTP processing if you want to have this scope provide an address for any BOOTP client that requests one. If you do not enable dynamic BOOTP, you must make reservations for each BOOTP client for which you want this scope to provide an address.

# Enabling BOOTP for Scopes

You can enable BOOTP processing for a scope. Set certain attributes and BOOTP reply options for a created policy in the local cluster web UI, or use **policy** *name* **create** [*attribute=value*] and **policy** *name* **set** *attribute=value* [*attribute=value* ...] in the CLI, to configure BOOTP. Set the policy attributes and options as a comma-separated list. The attributes are entities to use in a client boot process:

- *packet-siaddr*—IP address of the next server

- *packet-file-name*—Name of the boot file
- *packet-server-name*—Hostname of the server

The server looks through the policy hierarchy for the first instances of these attribute values.

The CLI **policy** *name* **setOption** <*opt-name* | *id*> *value* [**-blob**] [**-roundrobin**] requires spaces (not equal signs) before values. When **-roundrobin** is enabled, instructs the DHCP server to return option data which contains more than one value in a different, rotated, order. A particular client will always get the same order, but different clients will get different "rotations" of the order of the multiple values configured for the option based on their client identifier.

Also, enable BOOTP and dynamic BOOTP, if desired, and ensure that the DHCP server updates the DNS server with BOOTP requests. The options are:

- Set the option *dhcp-lease-time*.
- Enable the *dynamic-bootp* attribute.
- Enable the *update-dns-for-bootp* attribute.
- Enable the *update-dns-for-bootp* attribute.

# Moving or Decommissioning BOOTP Clients

When you move or decommission a BOOTP client, you can reuse its lease. To decommission a BOOTP client, you must remove its lease reservation from the scope and force its lease to be available.

Force the lease to be available in the local cluster web UI, or use **scope** *name* **removeReservation** (*ipaddr* | *macaddr* | *lookup-key*) [**-mac** | **-blob** | **-string**] and **lease** [*vpn-name/*]*ipaddr* **force-available** in the CLI.

# Using Dynamic BOOTP

When you use dynamic BOOTP, there are additional restrictions placed on the address usage in scopes, because BOOTP clients are allocated IP addresses permanently and receive leases that never expire.

If you are using DHCP failover, when a server whose scope does not have the *dynamic-bootp* option enabled goes into PARTNER-DOWN state, it can allocate any available IP address from that scope, no matter whether it was initially available to the main or backup server. However, when the *dynamic-bootp* option is enabled, the main server and backup servers can only allocate their own addresses. Consequently scopes that enable the *dynamic-bootp* option require more addresses to support failover.

When using dynamic BOOTP:

1. Segregate dynamic BOOTP clients to a single scope. Disable DHCP clients from using that scope. In the local cluster web UI, under the BOOTP attributes for the scope, disable the *dhcp* attribute. In the CLI, use **scope** *name* **disable dhcp**.
2. If you are using DHCP failover, set the *failover-dynamic-bootp-backup-percentage* attribute for the DHCP server to allocate a greater percentage of addresses to the backup server for this scope. This percentage can be as much as 50 percent higher than a regular backup percentage.

# BOOTP Relay

Any router that supports BOOTP relay usually has an address that points to the DHCP server. For example, if you are using a Cisco router, it uses the term *IP helper-address*, which contains an address for a specific machine. In this case, use this address to forward all BOOTP (and therefore DHCP) broadcast packets. Be sure that you configure this address on the router closest to your host.

**Tip** If your DHCP clients are not receiving addresses from the DHCP server, check the network configuration, particularly the router or relay agent configuration, to verify that your network devices are set up to point to your Cisco Prime Network Registrar DHCP server address.